BIBLIOTECA BASICA

INFORMATICA

CONSTRUYA SU PROPIA BASE DE DATOS

40



INGELEK

BIBLIOTECA BASICA INFORMATICA

CONSTRUYA SU PROPIA BASE DE DATOS

40

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Redactor técnico:
Enrique Monsalve

Colaboradores:
Casimiro Zaragoza

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-66-7
ISBN de la obra: 84-85831-31-4
Fotocomposición Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-11.988-1986
Precio en Canarias, Ceuta y Melilla: 380 pts.

INDICE

PROLOGO

El ritmo de vida aumenta, la actividad se hace frenética y se acumulan los papeles. El médico se percata de que desperdicia su propia profesionalidad compilando impresos, actualizando fichas y reconstruyendo historias clínicas. El abogado se siente atrapado. El director necesita informaciones concisas en tiempo real, sin tener que interpretar personalmente números, datos y tablas. El empleado se ocupa del trabajo impersonal ligado a la correspondencia. Se hace cada vez más difícil gestionar montañas de datos, laberintos de informaciones y se siente una necesidad cada vez mayor de delegar un trabajo tan necesario, pero enojosamente repetitivo, al que sepa hacerlo rápido y bien. Una ayuda válida existe ya en la actualidad y es la que proporciona el ordenador, una máquina lógicamente un tanto sofisticada y en evolución, pero que no puede ignorarse.

Cada aplicación del ordenador con respecto a las problemáticas de gestión gravitan en torno a una organización de datos. Las bases de datos, esas desconocidas, podemos imaginarlas como un criterio de organización de la información y una metodología de utilización, así como en una organización de transportes aéreos existen los medios y las rutas que permiten su empleo. Este elemento fundamental de la informática fue y es objeto de profundas investigaciones, algunas en el ámbito del proceso electrónico de datos, debido a la gran demanda en el mercado. Ya se ha hecho mucho a este respecto y cada ordenador construido para aplicaciones de gestión está organizado de modo que facilita la tarea del proyectista de bancos de datos, tanto desde el punto de vista estructural para el acceso a las memorias de masa como por el software de base especializado para estas aplicaciones.

Pero ¿qué se puede hacer con un ordenador personal? ¿Qué permiten los límites naturales de memoria y cuáles de los conceptos y soluciones técnicas adoptados en el ordenador de nivel superior se pueden aplicar? Este es el núcleo fundamental de nuestro libro, que se propone aclarar, en la medida de lo posible, tales conceptos con un ejemplo de base de datos para su ordenador personal. Antes de ello, consideramos necesario "refrescar" algunos principios teóricos de la organización de datos y reflexionar sobre cuáles pueden ser los escollos consiguientes en el ámbito del ordenador personal.

No pretendemos difundir un texto de opinión y consideramos más importante el aspecto instructivo que lo que una realización práctica puede ofrecer al lector.

Con este volumen finaliza la Biblioteca Básica Informática. Queremos agradecerles a todos ustedes el apoyo y el aliento que siempre nos han ofrecido. Esperamos haberles correspondido adecuadamente y seguirles encontrando en las demás publicaciones de Editorial Ingelek.

CAPITULO I

UN POCO DE TEORIA

El ordenador está constituido por tres partes principales: un instrumento lógico (CPU o unidad central), la memoria con la que trabaja y los dispositivos a través de los cuales se comunica con el mundo exterior. La memoria con posibilidades de acceso directo desde la CPU está limitada por motivos de costes y por el tipo de unidad central. No obstante, se dispone de memorias de masa de gran capacidad para superar esta limitación, tales como discos, cintas magnéticas, etc. En tal caso los accesos son mucho más lentos, porque el tratamiento sólo es posible transportando primero los datos desde estas memorias auxiliares a la memoria interna. De aquí nace la problemática de la gestión de datos, que consiste en encontrar la solución de compromiso entre costes, tiempos y metodología.

Comenzamos por indicar cuáles son los principales problemas planteados en el proyecto de un banco de datos, en los casos más generales, y a continuación veremos cómo aplicar el estudio teórico a nuestro caso, esto es, a la aplicación en el ordenador personal. El primer problema que se plantea es cómo organizar los datos, es decir, qué tipo de estructura lógico-física se adapta mejor a las exigencias que nuestro archivo trata de superar. La estructura física caracteriza el modo en el que los datos están físicamente representados en la memoria del ordenador. Por el contrario, la estructura lógica, con independencia de la estructura física de almacenamiento, constituye la representación de los datos tal como se "ven" por el usuario del sistema.

Las informaciones que almacenamos en un archivo pueden considerarse como grupos de informaciones de base, entre las

cuales existen relaciones lógicas que las ponen en correlación. La estructura lógica de un archivo indica cuáles son los datos elementales y sus grupos, cuáles son las relaciones entre datos para formar los grupos y cuáles son las relaciones entre los grupos que forman el archivo.

Mucho más complejo es lo que se refiere a la estructura física. La elección de esta última depende, sobre todo, del hardware de que se disponga, del tipo de proceso que se quiere desarrollar y de los tiempos de acceso a la información necesarios para que esta última no pierda validez.

Los parámetros principales para la elección de una estructura física en lugar de otra son:

- El coeficiente de densidad, dado por la relación entre el espacio de memoria ocupado por las informaciones y el espacio total del archivo (incluyendo también las informaciones complementarias para la extracción de los datos principales).
- La longitud de búsqueda, es decir, el número de accesos necesarios para obtener una información particular.
- facilidad de actualización del archivo: eliminación, inserción o modificación de un dato.

Los tipos más sencillo de estructura física son tres: secuencial, aleatoria y de listas. A continuación se expondrán las ventajas y los inconvenientes y más adelante trataremos de estructuras de tipo más complejo.

Estructuras secuenciales

Este es uno de los métodos más conocidos y utilizados para la organización de los datos; su característica principal es que los datos que componen el archivo están almacenados de forma secuencial uno tras otro. Si consideramos la información subdividida en dos partes: **atributo identificador o clave** (código asociado a cada información que permite su referencia) y la **información propiamente dicha,** tendremos una sucesión ininterrumpida de parejas (véase figura 1).



Figura 1.—Estructura secuencial.

El problema de la búsqueda de una información particular puede resolverse de varios modos:

- Búsqueda completa; se tiene acceso a cada registro, mediante una exploración, hasta encontrar la clave deseada o hasta recorrer todo el archivo. La longitud media de búsqueda, si N es el número de informaciones contenidas en el archivo, es (N+1)/2.
- Búsqueda en serie; se exige que el archivo esté ordenado por valores crecientes de la clave y se accede de forma secuencial a los datos hasta encontrar la clave buscada o bien una clave mayor que ella. La longitud media de búsqueda es N/2.
- Búsqueda binaria; también en este caso se requiere que el archivo esté ordenado. La búsqueda se realiza por sucesivas bisecciones del archivo, por lo que a cada paso se actúa sobre un número de datos igual a la mitad del paso anterior. Se comienza por comparar la clave buscada con la que se encuentra a la mitad del archivo; si ésta es mayor se desecha la segunda mitad del archivo y si es menor se desecha la primera mitad. En el segundo paso se busca la clave que se encuentra a la mitad de la nueva parte de datos y se procede como en el paso anterior hasta encontrar o no el elemento. Este método es muy rápido y eficiente porque el número máximo de accesos para encontrar una clave, o para establecer que no existe, es $Log_2(N)$; por ejemplo, para 1.024 registros el número máximo es 10.

La estructura secuencial presenta unas notables ventajas de sencillez de gestión y de ocupación de memoria, no exigiendo informaciones suplementarias para la búsqueda del dato. Con la búsqueda binaria se tiene también un acceso muy rápido, pero las operaciones de actualización son algo complejas, por cuanto la introducción de un nuevo registro exige una reordenación de todo el archivo o una fusión de los nuevos datos. Existen, sin embargo, varios algoritmos muy eficaces de ordenación (BUBLESORT, HEAPSORT, QUICKSORT, etc.).

Estructuras aleatorias

Las estructuras aleatorias, es decir, aquellas en las que no existe relación entre las direcciones de los diversos datos, obtienen directamente, a partir de las claves de acceso, las informaciones sobre la **dirección** del elemento.

Existe, pues, una "función de acceso" que permite obtener la dirección de la clave. Hay varios métodos de correlación:

- Método de diccionario: permite el acceso al registro mediante la previa lectura de la dirección en una tabla; es decir, se crea una matriz de correspondencia entre clave y dirección. Se busca la clave en la tabla y se extrae también la dirección de la información correspondiente (véase figura 2).
  También es posible la subdivisión del archivo en varios subarchivos, con el almacenamiento de la clave más alta de cada subarchivo. La búsqueda se realiza primero en el diccionario, en el que se busca el subarchivo de pertenencia del dato, y luego en el subarchivo mediante uno de los métodos vistos para las estructuras secuenciales.
- Método de cálculo o direccionamiento Hash: permite la utilización de archivos de acceso directo; la asociación clave-dirección se realiza mediante un algoritmo que extrae de la propia clave la dirección del dato correspondiente. El inconveniente de este método es encontrar un algoritmo que proporcione un resultado uniformemente distribuido, de modo que no se produzca una superposición de registros y que no se tengan grandes zonas vacías.

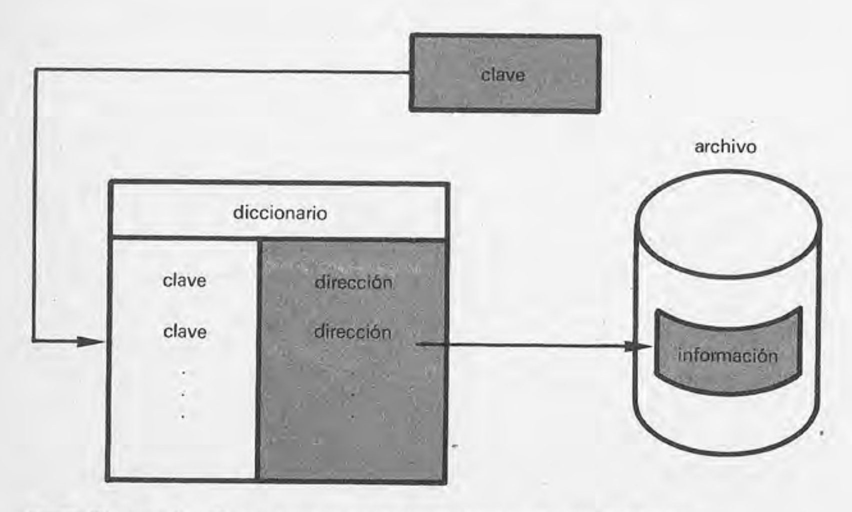


Figura 2.—Estructura aleatoria; método del diccionario.

Algunos algoritmos HASH se basan, por ejemplo, en: la dirección viene dada por un subconjunto de bits de la clave; la dirección es la suma, bit a bit, de diversos subconjuntos de bits de la clave y la dirección es un subconjunto de bits del cuadrado de la clave.

Una vez elegido el algoritmo de cálculo se trata de elegir un algoritmo de exploración del área direccionada para controlar mejor la eventualidad de que dos o más claves den como resultado la misma dirección. Los más difundidos son los de:

- Exploración lineal, caracterizada porque en caso de colisión se explora el archivo a partir de la dirección encontrada de forma secuencial con un cierto paso >=1.
- Exploración pseudoaleatoria, caracterizada porque los incrementos de dirección con respecto al generado por el algoritmo no son constantes, sino que son producidos por una subrutina que genera números enteros de manera aleatoria.
- Exploración cuadrática, es un método similar al de la exploración lineal, con la salvedad de que después de cada colisión para una clave el incremento se aumenta en 1 (es decir, a la primera colisión se tendrá un incremento de 1, a la segunda de 2, etc.). La variación de este método para aumentar su eficacia es la del cociente cuadrático, que consiste en hacer variar el incremento con la clave, en lugar de que sea constante.
- Método de concatenación, mediante el cual, una vez encontrada la dirección con el algoritmo, si está libre se almacena el dato y si no lo está se construye una cadena (cola) de elementos correspondientes a dicha dirección. A continuación veremos qué son las cadenas.

Estructuras de lista

En la estructura de lista, los registros están unidos entre sí mediante **punteros** que, en cada elemento, indican la dirección del sucesivo (véase figura 3).

La lista tiene una cabecera que apunta al primer elemento y cada elemento apunta al siguiente, que puede estar ubicado en cualquier parte de la memoria (de masa o interna del ordenador) y no necesariamente en las proximidades, mientras que el último elemento tiene el puntero igual a 0.

Las actualizaciones de una lista son bastante sencillas:

- Inserción de un elemento X en una lista (figura 4). Se puso el ejemplo de la inserción en la cabecera de la lista porque

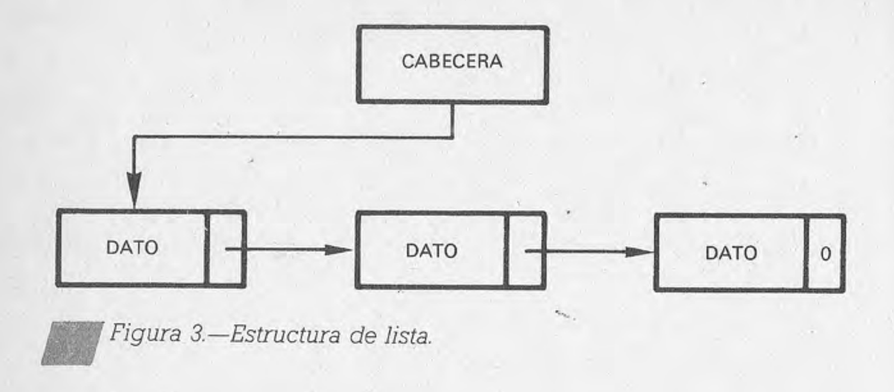


Figura 3.—Estructura de lista.

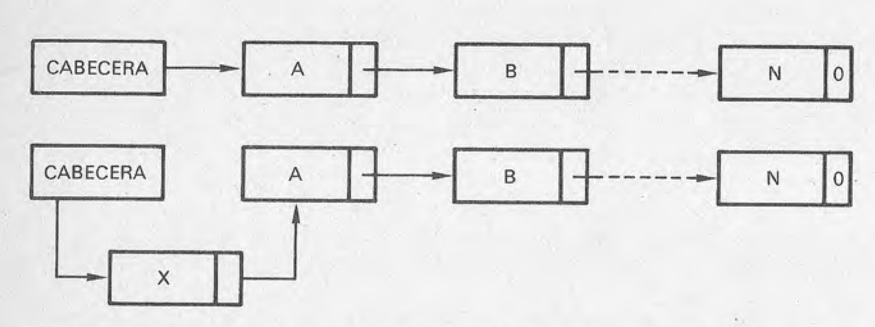


Figura 4.—Una lista antes y después de la inserción del elemento X.

es más rápida, pero también una inserción en cualquier otro punto funciona del mismo modo.

- Borrado de un elemento X de una lista (figura 5).
  Una posibilidad adicional de la estructura de lista es la lista bidireccional, en la que cada elemento contiene dos punteros, uno al elemento siguiente y otro al anterior (véase figura 6).

La búsqueda de un registro particular se realiza recorriendo la lista de la cabecera hasta encontrar el elemento deseado. Para el control de los elementos borrados y de los elementos libres para poder utilizarlos en las inserciones hay dos métodos principales:

- Lista libre: es una lista formada por elementos vacios; cuando se tiene necesidad de un elemento para insertar un registro, se le toma de la lista libre y cada elemento suprimido se inserta en ella de modo que se pueda reutilizar.

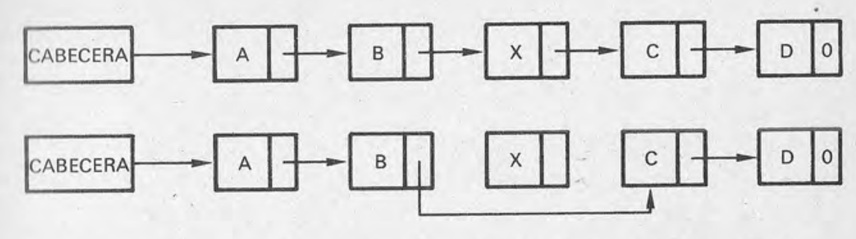


Figura 5.—Una lista antes y después de eliminar el elemento X.

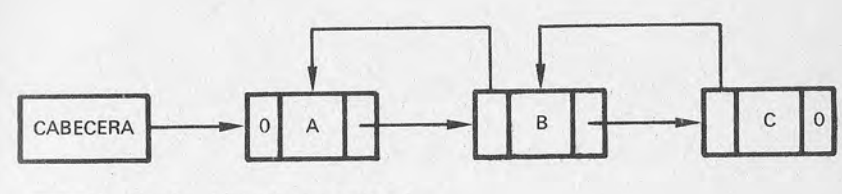


Figura 6.—Lista bidireccional.

- Garbage collection ("recogida de desechos"): los elementos borrados no se tienen en cuenta y, de forma periódica, un programa examina toda la memoria del archivo y reúne estos elementos no utilizados en una lista.
  La estructura de lista tiene el defecto de que se necesita memoria suplementaria para los punteros y de que el tiempo de búsqueda depende, en gran medida, de la longitud de la lista.

Estructuras más complejas

Para los problemas que no tienen una solución eficiente con las estructuras hasta ahora vistas, se han estudiado otras estructuras más complejas.

- Estructuras de listas invertidas: si se quisieran recoger grupos de informaciones en listas ordenadas según diversos parámetros (véase figura 7) se tendría un despilfarro de me-

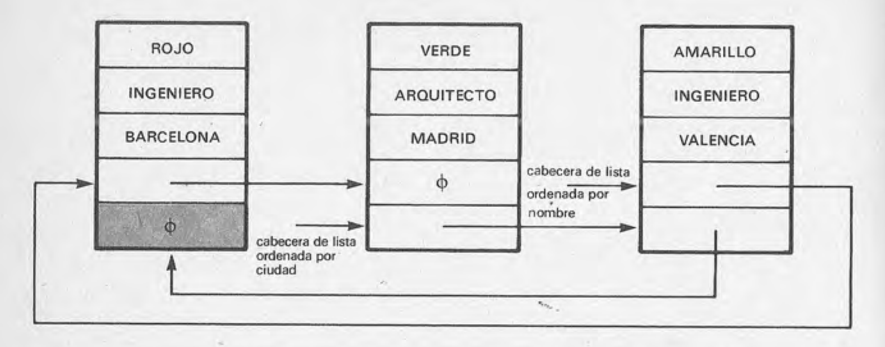


Figura 7.—Informaciones relacionadas en listas ordenadas según algunos atributos.

moria por cuanto que sería necesario un puntero por cada tipo de lista. Un método mejor es el de definir por cada elemento (en el ejemplo, por cada persona) un conjunto de atributos (título de estudio y ciudad) y se crea luego un **índice** (véase figura 8) que contiene la identificación de los atributos y los valores correspondientes a cada elemento que tiene ese atributo, de modo que cada elemento se reduzca a un registro único en una determinada dirección.

INDICE	
ATRIBUTO	DIRECCIONES
INGENIERO	133,53
ARQUITECTO	74
MADRID	74
VALENCIA	133
BARCELONA	53

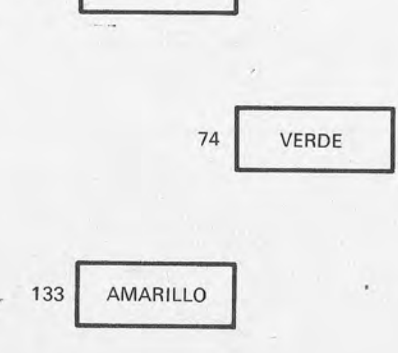


Figura 8.—Estructuras de listas invertidas.

En este punto es posible realizar una búsqueda de los registros según uno cualquiera de los atributos definidos, con el empleo de punteros introducidos en la tabla de índice. La ventaja de las listas invertidas es que permiten tener acceso a todos los datos con la misma facilidad y con una mayor rapidez, por cuanto que se efectúan las búsquedas solamente en los índices y luego se realiza un acceso directo al registro buscado.

- Estructuras en anillo: se trata de una ampliación de las listas simples en las que el último elemento, en lugar de tener un puntero nulo, apunta al primer elemento de la lista. Permiten también ramificaciones de cualquier registro para individualizar otros datos en correlación con el mismo (figura 9), con la construcción de las denominadas estructuras **jerárquicas.** Estas estructuras permiten representar relaciones complejas y la búsqueda de los datos a partir de cualquier punto del archivo.
- Estructuras de listas múltiples: estas estructuras recuerdan mucho a las invertidas, con la salvedad de que en correspondencia con los atributos se almacena solamente la dirección de comienzo de la sublista y su longitud; además,

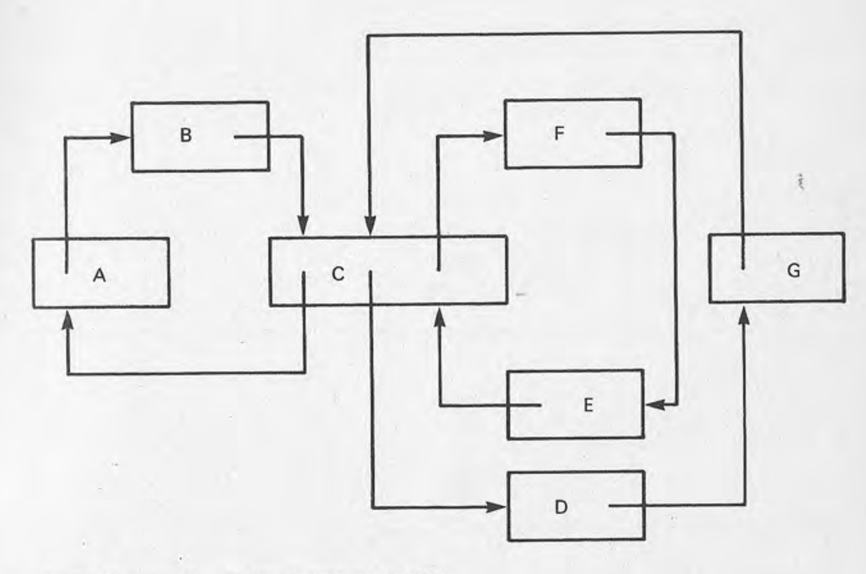


Figura 9.—Estructuras en anillo.

la memoria en la que están registradas las sublistas está subdividida en bloques (figura 10). Cada registro está direccionado por el número del bloque y por la dirección en su interior. Cuando se busca un elemento se recupera con un solo acceso el bloque completo, por lo que se puede ahorrar tiempo y, por consiguiente, es preferible almacenar los datos, en la medida que sea posible, en un mismo bloque. En la figura 10 se muestra un ejemplo en el que las listas correspondientes a los atributos INGENIERO y BARCELONA son 4 y 3 respectivamente y comienzan una de ellas en el bloque 0 con el elemento 3 y la otra en el bloque 0 pero con el elemento 5. Además, se ve que pueden existir elementos como el 7 que sean comunes a varias listas. Las listas múltiples, con respecto a las listas invertidas, tienen la ventaja de ser de más fácil actualización y programación, pero traen consigo tiempos más largos de búsqueda debido al recorrido de las listas.

- Estructuras en árbol: son de gran utilidad cuando se necesita que los datos estén organizados de modo jerárquico.

INDICE
ATRIBUTO
INGENIERO
BARCELONA
0,3 / 4
0,5 / 3
BLOQUE 0
BLOQUE 1
3
7
5
11
12
6

Figura 10.—Estructuras de listas múltiples.

Un árbol puede definirse como un gráfico conexo provisto de circuitos cerrados y constituido por un cierto número de nudos o nodos (las informaciones) conectados por líneas, con la propiedad de que dos nodos cualesquiera están unidos por un solo camino y de que un árbol con N nodos contiene N-1 líneas.
El primer nudo se denomina **raíz.** Un árbol se llama binario si a partir de cualquier nudo salen, como máximo, dos líneas o ramas (figura 11). Cada nudo en un árbol binario puede representarse como el conjunto del dato junto con los punteros de las dos ramas derecha e izquierda (figura 12). El recorrido de un árbol binario ("vista de un árbol") puede realizarse de tres modos:

— Orden previsto: examen del nudo raíz, examen del subárbol de la izquierda y examen del subárbol de la derecha (en el ejemplo, el orden sería ABDECFGH);
— Orden simétrico: examen del subárbol de la izquierda, examen del nudo raíz y examen del subárbol de la derecha (en el ejemplo, DBEAFCHG);
— Orden diferido: examen del subárbol de la izquierda, examen del subárbol de la derecha y examen de la raíz (en el ejemplo, DEBFCHGA).

Una estructura en árbol binario tiene la propiedad de que el subárbol izquierdo de cada nudo contiene solamente datos cuya

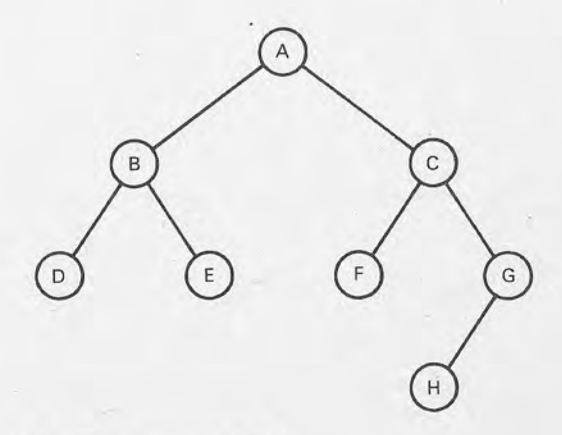


Figura 11.—Estructura en árbol.

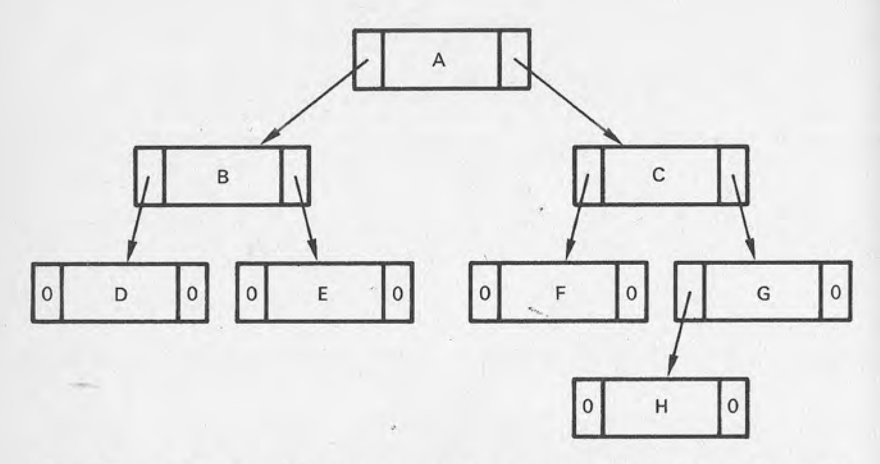


Figura 12.—Posible representación interna del árbol de la figura 11.

clave es inferior a la del dato de dicho nudo, mientras que el subárbol derecho contiene solamente claves mayores. La búsqueda o el almacenamiento es bastante simple, por cuanto que en cada nudo resulta inmediata la elección del camino a seguir hasta encontrar el dato o una rama libre en donde almacenarlo.

CAPITULO II

APLICACION A UN ORDENADOR PERSONAL

Un ordenador personal está constituido por unos componentes que tienen las mismas funciones lógicas que en un ordenador tradicional, es decir, unidad central, dispositivos de Entrada/Salida, (E/S o, en inglés, I/O) y memoria, en parte de acceso directo (RAM: Random Access Memory) y en parte de masa. Los costes moderados y el empleo al que está dirigido el ordenador personal han orientado a los fabricantes a la elección de CPUs de 8 bits generalmente. Por consiguiente, la memoria directamente direccionable varía entre 4 y 64 Kbytes, por lo que considerando que una parte de esta capacidad de memoria es utilizada para funciones internas (intérprete, sistema operativo, etc.), no queda memoria suficiente para tratar el número de informaciones necesarias para una base de datos en RAM, salvo para archivos mínimos.

Es pues inevitable la necesidad de emplear soportes de memoria externos, los cuales habrán de tener un acceso relativamente rápido y, sobre todo, una lógica de control que deberá permitir el acceso directo a la información elemental puesto que, de no ser así, las lecturas secuenciales tendrían un efecto desfavorable sobre los tiempos. No consideramos, pues, la hipótesis de utilizar un soporte secuencial tal como las cintas magnéticas, sino solamente los **discos flexibles,** que consideramos los soportes más válidos para una aplicación informativa ligada con la base de datos. En la mayor parte de los ordenadores personales, las unidades de disco pueden admitir entre 180 y 500 Kbytes en línea, es decir, con toda la información con posibilidad de acceso directo sin intervención manual para cambio de disco.

Volvamos en este punto a los conceptos teóricos para constatar cuánto y con qué resultados son aplicables a los ordenadores personales los métodos anteriores.

- Estructura **secuencial:** al ser la más simple sería aparentemente la más adecuada para nuestro caso; sin embargo, al realizar informaciones con longitud variable, con el consiguiente empaquetamiento de los datos, los tiempos de acceso hacen pesado el tratamiento y las exploraciones prolongadas dan lugar a mayores solicitudes de la unidad de masa. Estructuras de dicho tipo son preferibles para datos utilizables en bloques, es decir, una serie de datos que sean objeto de lectura, tratamiento y escritura siempre en conjunto.
- Estructura **aleatoria:** Para los ordenadores personales que permiten el acceso directo al registro en disco, esta solución resulta óptima por cuanto que el proceso necesario para obtener la dirección de la clave no es problemático en absoluto y el acceso a una información individual se hace casi inmediato. El método del diccionario exige mayor espacio en disco para el almacenamiento de los pares clave-dirección, que pueden estar todos ellos en memoria o bien en disco en el momento de su proceso. En el primer caso resulta necesario cargar todo el diccionario en memoria al comienzo, con lo que utiliza los tiempos de las búsquedas sucesivas; en el segundo caso se puede efectuar una búsqueda binaria directamente en el disco, manteniendo el diccionario ordenado por claves. Esta organización, que es muy cómoda en las fases de búsqueda y modificación de la información, es un poco más honerosa durante las introducciones y borrados, porque se necesitará en cada ocasión la reestructuración del diccionario. El método HASH, aunque sea algo problemático en el tratamiento de los bits para el cálculo de la dirección en BASIC, es bastante eficaz en lo que respecta a los tiempos de acceso cuando el archivo no está llenado en más del 75 por cien. Especialmente simplificadas resultan todas las fases de gestión, pero tendrá una utilización deficiente la memoria de masa.
- Estructuras de **listas:** para las aplicaciones con ordenador personal representan una sofisticación, pero pueden resultar de utilidad y eficacia si se conectan a una estructura base de otro tipo, por cuanto permiten un ahorro de memoria y una mayor sencillez de gestión. Por ejemplo, en la elaboración de un archivo médico resulta posible asociar a una serie de datos personales una lista de longitud variable, según el paciente de que se trate, que contenga los datos correspondientes a las diversas visitas (figura 1). Los punteros pueden introducirse bien sea físicamente en el interior del registro, bien por separado en el fichero correspondiente, bien conectados de forma lógica al que contiene las listas, como se realizará en el ejemplo propuesto.
- Estructuras más complejas: aunque sea posible su realización no resultan recomendables para las aplicaciones prácticas en un ordenador personal. Ello es así porque el aumento de dificultad en la gestión de los datos no está compensada por un suficiente incremento de eficacia. Además, los bancos de datos realizables no pueden tener una dimensión que justifique una sofisticación de esta naturaleza.

Definición de la aplicación

El ejemplo de aplicación ha de ser de uso generalizado y que satisfaga las exigencias más concretas con un solo programa, estudiado de modo que sea flexible y adaptable.

Consideremos algunas de las exigencias típicas:

- El archivo de las historias clínicas consiste en la estructuración de un fichero, constituido por grupos de datos correspondientes a cada paciente de los que no se conoce a priori su longitud, por cuanto que la historia clínica puede ser variable. El médico tiene necesidad de un instrumento que le permita:

  1) introducir nuevos pacientes
  2) actualizar la situación
  3) conocer todos los datos patológicos
  4) extraer relaciones y situaciones siempre actualizadas.

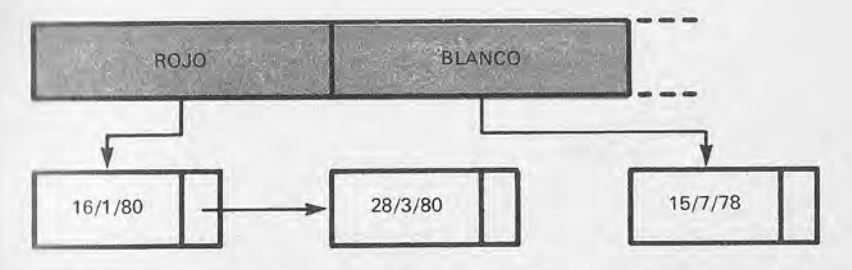


Figura 1.—Ejemplo de archivo de datos fijos, con listas conectadas a variables de datos.

- Para la gestión de ventas con agentes es preciso conocer, además de las informaciones generales sobre los volúmenes de venta, informaciones más detalladas relativas a cada representante. Asimismo, podría ser interesante conocer estadísticas de venta por representante y por artículo.
- La gestión de los contactos con posibles clientes prevé la definición de las características personales y la posibilidad de seguir el desarrollo de las negociaciones y de la adquisición de pedidos.

Algo similar podría aplicarse a la gestión de órdenes, pedidos, etc. Todas las aplicaciones ilustradas tienen en común el hecho de que las informaciones están constituidas por una parte fija o de base, a la que se agregan informaciones suplementarias. En las fichas clínicas los datos base son las informaciones personales y los datos suplementarios, o auxiliares, los resultados de las diversas visitas.

Decidimos, pues, la conveniencia de trabajar en un archivo cuya estructura lógica esté constituida por un conjunto de grupos de datos principales a los cuales se pueden asociar cantidades variables de otras informaciones.

Estudio de viabilidad

Para poder realizar un estudio de viabilidad del programa es necesario coocer el sistema con el que trabajamos.

El EUROPLUS de Apple II está provisto de un intérprete Basic de coma flotante en ROM y de hasta 48 Kbytes de memoria RAM, en donde reside el programa del usuario y, si se utilizan los discos flexibles, el sistema operativo de discos DOS que ocupa 10 Kbytes. Hay posibilidades de conexión de hasta 12 unidades "floppy", cada una de las cuales puede contener 116 Kbytes. Al ser el sistema operativo residente en cada disquete, se dispone de 100 Kbytes para su utilización.

Lo anterior nos impone una limitación en el número de registros almacenables en cada disco; al no querer (aunque sea posible) gestionar ficheros multivolumen, estructuraremos el archivo base en un solo disquete y, por consiguiente, el número de registros suceptibles de gestión será 100 K/longitud del registro. Al tener la posibilidad de acceder en modo directo a los registros, por número de orden, podemos utilizar para el archivo base la estructura aleatoria con el método del diccionario, construyendo un índice que contiene todas las claves de acceso y la dirección correspondiente, dada como número del registro. Además, es posible la realización de una estructura de listas y podemas construir una o varias listas que terminen en un registro base.

El DOS 3.2 permite la utilización de fichero, sin tener que definir las características físicas (tales como longitud de registro, espacio, etc.), por lo que resulta sencillo emplear ficheros de descripción de la "personalización" y del estado del sistema.

Esta asignación dinámica del espacio nos permite una notable flexibilidad de configuración y podremos elegir entre diversas soluciones.

La estructura global (figura 2) constará de:

- Un archivo base, que contiene datos físicos no ordenados.
- Un índice principal, que contiene las claves y las correspondientes direcciones de los registros base. Este fichero deberá mantenerse ordenado por claves crecientes.
- Ocasionales índices secundarios para permitir el acceso a los registros base según otros campos.
- Uno o varios archivos organizados en listas con conexión lógica a los registros base.

Según las exigencias particulares, bastará "personalizar" la configuración, eligiendo el número de unidad a utilizar. Así podemos, en caso de que tengamos pocos registros base y no quera-

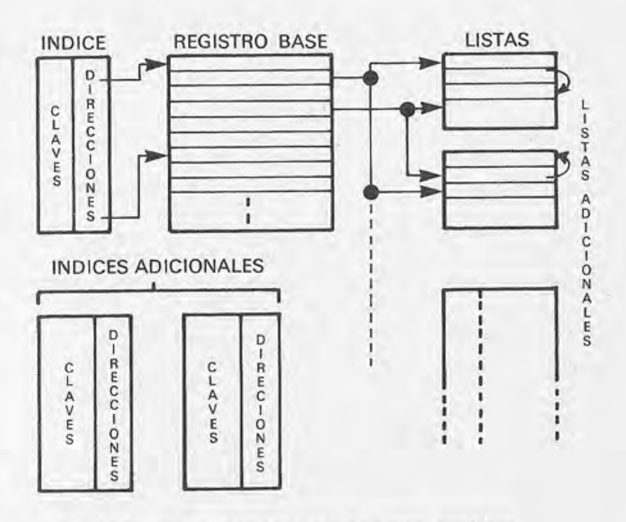


Figura 2.—Estructura general de nuestro archivo.

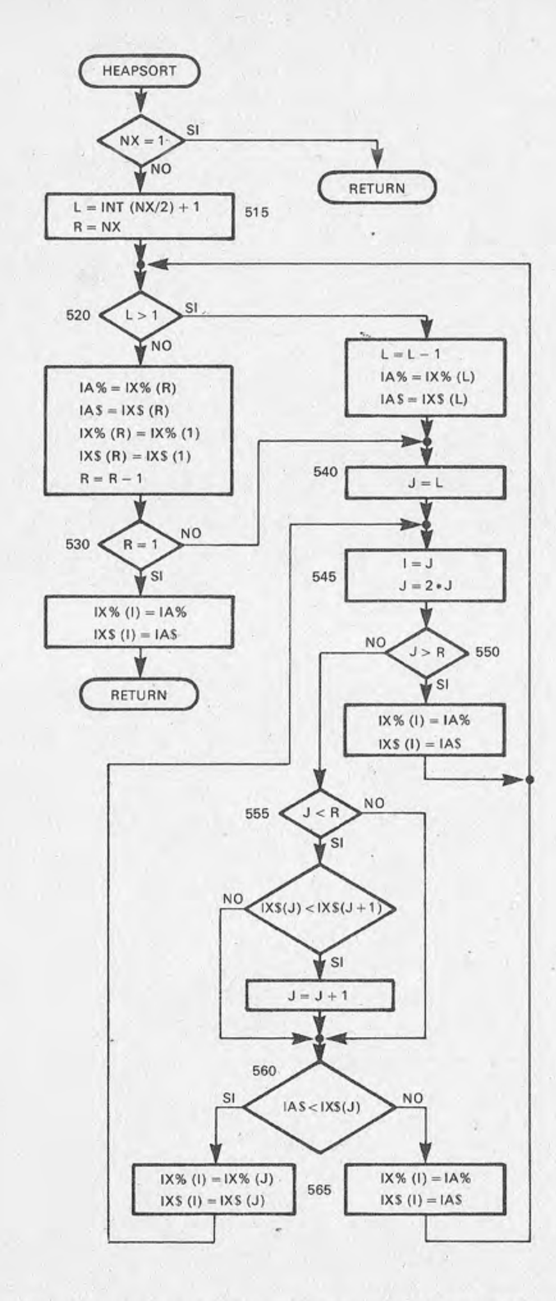


Figura 3.—Diagrama de flujo de la rutina HEAPSORT. Los números junto a algunos bloques se refieren a las líneas asociadas del listado.

mos índices secundarios ni listas, ubicar todo el archivo en un solo disquete. Para mayores necesidades, se podrá reservar el primer disco para el archivo base y el índice principal, mientras que las listas y los índices secundarios irán en el segundo disco. Asimismo, los índices secundarios podrán ser objeto de gestión con el empleo de una tercera unidad. El hecho de tener las listas en una unidad especializada permite disponer de más grupos de listas por cada registro base, con el simple cambio del disquete de la unidad. Por ejemplo, es posible asociar al mismo archivo base unas listas mensuales.

Rutinas de utilidad práctica

En el curso del estudio técnico y de viabilidad, se hizo alusión a la necesidad de **ordenación** y de **búsqueda binaria.** Ahora hacemos mención de las rutinas que ordenan un vector de N elementos y que realizan la búsqueda binaria en tal vector. Hemos elegido dos rutinas cuyos algoritmos fundamentales son debidos al Knuth.

Heapsort (figuras 3 y 4) es uno de los algoritmos más rápidos; tiene la característica de emplear un tiempo casi constante cualquiera que sea la configuración inicial del vector. El proceso se subdivide en dos partes:

```
1 REM ***************************
2 REM * SUBRUTINA DE ORDENACION *
3 REM *         EJEMPLO         *
4 REM ***************************
510 IF NX=1 THEN RETURN
515 L=INT(NX/2)+1:R=NX
520 IF L>1 THEN L=L-1:PA=PI%(L):GOTO 540
525 PA=PI%(R):PI%(R)=PI%(1):R=R-1
530 IF R=1 THEN PI%(1)=PA:RETURN
540 J=L
545 I=J:J=2*J
550 IF J>R THEN PI%(I)=PA:GOTO 520
555 IF J<R THEN GOSUB 600
560 IF IX$(PA) < IX$(PI%(J)) THEN PI%(I)=PI%(J):GOTO 545
565 PI%(I)=PA:GOTO 520
```

Figura 4.—Listado de la rutina HEAPSORT.

- Una fase preparatoria rápida, en la que se construye un "heap" ("montón").
- Una fase sucesiva de ordenación propiamente dicha.

La rutina ordena un vector IX$ de claves alfanuméricas que contiene NX elementos, manteniendo la asociación con el correspondiente vector IX% de direcciones (enteras). El flujo no resulta estructurado, lo que propicia una mayor velocidad de ejecución y un mayor espacio de memoria.

El algoritmo de búsqueda binaria (figura 5), anteriormente descrito, actúa sobre el vector IX$, buscando en el mismo la clave suministrada en KM$ y dejando en F el resultado de la búsqueda y en P el índice del elemento, si existiera.

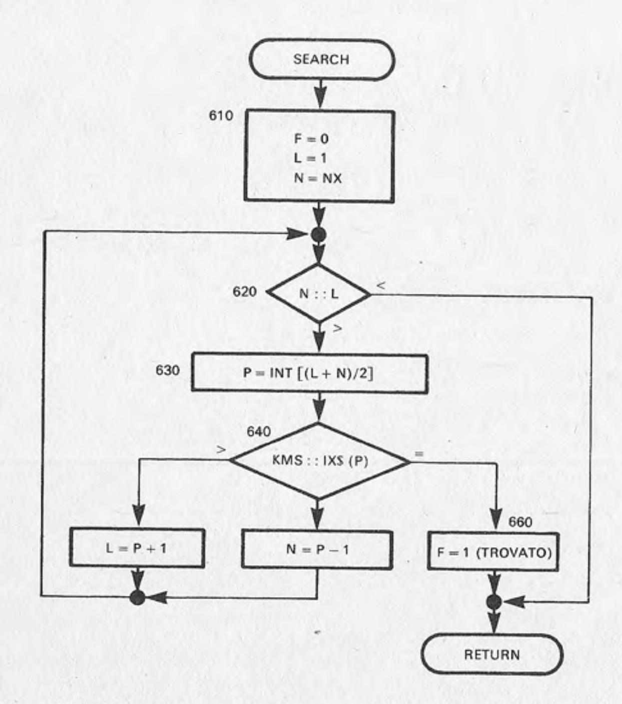


Figura 5.—Diagrama de flujo de la rutina por la búsqueda binaria. Los números se refieren a las líneas del listado asociadas.


```
600 REM ******************************
602 REM * SUBRUTINA BUSQUEDA BINARIA *
603 REM *          EJEMPLO           *
604 REM ******************************
610 F = 0: L = 1: N = NX
630 IF N < L THEN RETURN : REM NO ENCONTRADO
640 P = INT ((L + N) / 2)
650 IF KM$ < IX$ THEN N = P - 1:GOTO 630
660 IF KM$ > IX$ THEN L = P + 1:GOTO 630
670 F = 1:RETURN: REM ENCONTRADO
```

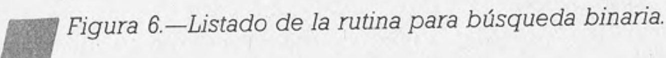


Figura 6.—Listado de la rutina para búsqueda binaria.

CAPITULO III

ARQUITECTURA DE LA BASE DE DATOS

Una vez conocidos los criterios generales de la estructura de datos, descritos en los dos primeros capítulos, vamos a profundizar en el estudio de la viabilidad para definir la arquitectura de la base de datos sobre la cual elaborar el programa de gestión. En este capítulo tendremos que ser necesariamente más prácticos y utilizar los instrumentos de hardware-software disponibles en el ordenador personal, en nuestro caso concreto del Appel II Plus. De la enojosa teoría pasamos al diseño lo que es ciertamente más estimulante.

El D.O.S. Apple

Además del lenguaje de programación BASIC es necesario conocer cuáles son las posibilidades funcionales del sistema Apple en el tratamiento de la información en los discos flexibles.

El sistema objeto de examen tiene una estructura de hardware flexible y ampliable, disponiendo de ocho ranuras ("slots") de salida no dedicadas a la conexión de periféricos específicos, la excepción son la ranura o slot 0, que está concebida para expansiones del lenguaje, y la número 7, reservada a la tarjeta PAL para visualización de colores. Las seis ranuras restantes pueden utilizarse indiferentemente para conectar impresoras, unidades para discos y otros tipos de periféricos. Cada interface para disco flexible puede controlar dos unidades. Para tener acceso a las informaciones residentes en un disco es necesario, en caso de que puedan detectarse ambigüedades, definir lo que se denomina ra-

nura y unidad de disco de pertenencia. Además, en un disco flexible pueden coexistir informaciones relativas a programas almacenados, zonas de memoria en formato binario y ficheros de datos de tipo secuencial o aleatorio, es decir, con subdivisión en registros de longitud fija. Las modalidades de tratamiento son diferentes para cada categoría; para nuestros fines, será suficiente conocer el tratamiento reservado a los programas Applesoft y a los ficheros de datos.

Cada grupo de informaciones, o ficheros, se identifica por un nombre de 30 caracteres como máximo y así se puede efectuar la llamada de programas o tener acceso a los archivos directamente por su nombre.

La gestión de los programas es especialmente sencilla: después de haber escrito el programa en la memoria del sistema mediante el teclado se le puede grabar escribiendo SAVE<nombre>, cargarlo desde el disco a la memoria del ordenador Apple escribiendo LOAD<nombre>, ejecutarlo escribiendo RUN<nombre> o simplemente RUN si el programa está ya en memoria.

Los ficheros de datos (texto) se dividen en dos categorías:

- Ficheros secuenciales, en los cuales las informaciones individuales están dispuestas de forma consecutiva y separadas por un carácter de fin de campo CR, Carriage Return (Retorno del carro) (figura 1).
  En tal caso, al tener que cambiar un dato por otro de longitud mayor, será necesario volver a escribir todo el fichero. Por el contrario, un fichero secuencial optimiza el espacio en disco y puede ser de utilidad para almacenar parámetros fijos, definidos en una ocasión y actualizados sólo raras veces.
- Ficheros aleatorios (file random), en los cuales el conjunto de datos está subdividido en registros de longitud fija que contienen uno o varios campos separados también por un retorno del carro (figura 2). Para actualizar un dato es suficiente volver a escribir el registro correspondiente; además, es posible tener acceso directo a un registro por su número, sin tener que explorar todos los datos de forma secuencial.

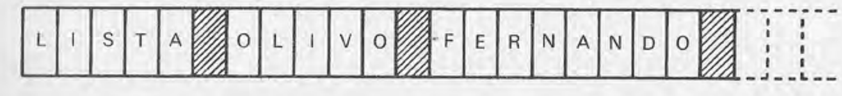


Figura 1.—Ejemplo de fichero secuencial.

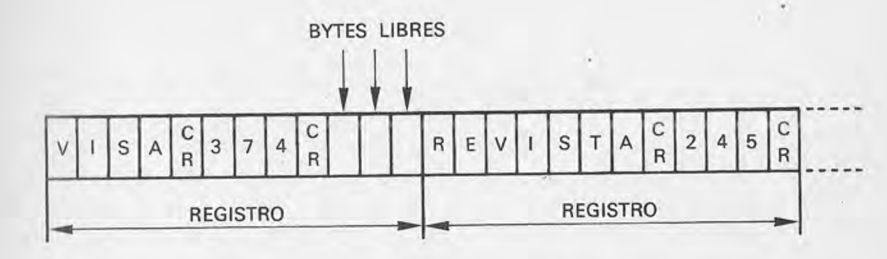


Figura 2.—Ejemplo de fichero aleatorio de 12 bytes.

Para el tratamiento de los ficheros de datos, definiremos cuatro funciones esenciales:

- **OPEN:** permite abrir un flujo informativo con un fichero residente físicamente en un cierto disco y, además, si la estructura del fichero es de tipo aleatorio, permite definir también la longitud del registro.
  Ejemplo:
  Secuencial OPEN<nombre>
  S<número ranura>
  D<número unidad>

  Aleatorio OPEN<nombre>
  L<long. reg.>
  S<número ranura>
  D<número unidad>

- **READ:** permite indicar a partir de qué fichero se efectuará la siguiente lectura de datos con la sentencia INPUT de BASIC. En el fichero secuencial la primera lectura (input) será la relativa al primer dato y las sucesivas se referirán a los datos consecutivos. En un fichero aleatorio el criterio es idéntico pero relativo al registro individual, es decir, es necesario definir en la función READ también el número del registro en que se ha de efectuar la lectura.
  Ejemplo:

  Secuencial READ<nombre>

  Aleatorio READ<nombre>
  R<núm. reg.>

- **WRITE:** permite identificar en qué fichero se efectuará la siguiente escritura de datos con la sentencia PRINT de BASIC. También en este caso las escrituras son consecutivas desde el comienzo del fichero o enésimo registro dependiendo del tipo de fichero secuencial o aleatorio.
  Ejemplo:

  Secuencial WRITE<nombre>

  Aleatorio WRITE<nombre>
  R<núm. reg.>

- **CLOSE:** permite cerrar un flujo informativo y asegura la escritura de las informaciones anteriores. En tal caso, no existen diferencias sintácticas entre ficheros secuenciales y aleatorios.
  Ejemplo:

  Secuencial CLOSE<nombre>

  Aleatorio CLOSE<nombre>

Todos los comandos de control de flujo antes citados, para ser identificados y ejecutados por el DOS (sistema operativo de control de disco) deberán aparecer como argumento de una sentencia PRINT precedidos por el carácter de control CONTROL-D, equivalente a CHR$(4). El sistema operativo de disco (DOS) de Apple permite muchos otros comandos de control, pero consideramos suficientes los cuatro descritos para la gestión de la base de datos objeto de examen; los lectores interesados por esta materia podrán encontrar una exposición exhaustiva en el manual DOS 3.2 Apple. Concluimos con un ejemplo completo de sentencia BASIC de control DOS:

```
10 PRINT CHR$(4)"WRITE PULPO,R12"
```

Estructuración de la base de datos

Como se comentó con anterioridad el objetivo que se persigue es organizar un archivo de base de datos accesible con una **clave preferencial** y, supongamos, cuatro **claves secundarias,** que pueda conectarse a un segundo archivo controlado de listas, en correlación lógica con los registros de base. Existen, pues, en un primer análisis, los archivos lógicos siguientes:

- Un archivo Base con el índice principal (MASTER INDEX) y los índices secundarios correspondientes.
- Un archivo de lista completo de los datos necesarios para la gestión.

Detengámonos en el archivo Base tratando de averiguar cómo puede estructurarse desde el punto de vista físico.

Todas las informaciones del archivo Base, es decir, los datos correspondientes a los diversos campos que constituyen los diferentes registros, pueden ser recogidas en un fichero que llamaremos MASTER FILE(MST.FL) de tipo obviamente aleatorio para facilitar las operaciones de escritura, lectura y modificación. La longitud del registro Base será, pues, la suma de las longitudes de los correspondientes campos con el correspondiente carácter CR de final de campo (figura 3).

Para hacer más rápidas las operaciones de acceso, introducción y eliminación del registro Base no mantendremos dicho fichero ordenado, sino que tendremos acceso al mismo utilizando un diccionario de claves-punteros que constituye un fichero MASTER INDEX (o MST.INX) que, a diferencia con el anterior, mantendremos siempre ordenado y presente tanto en la memoria como en disco (figura 4).

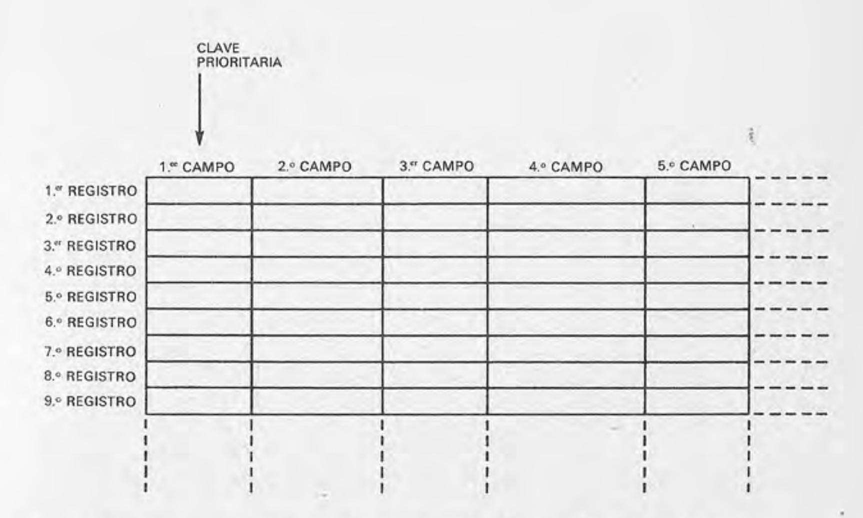


Figura 3.—Estructura aleatoria del fichero MASTER FILE.

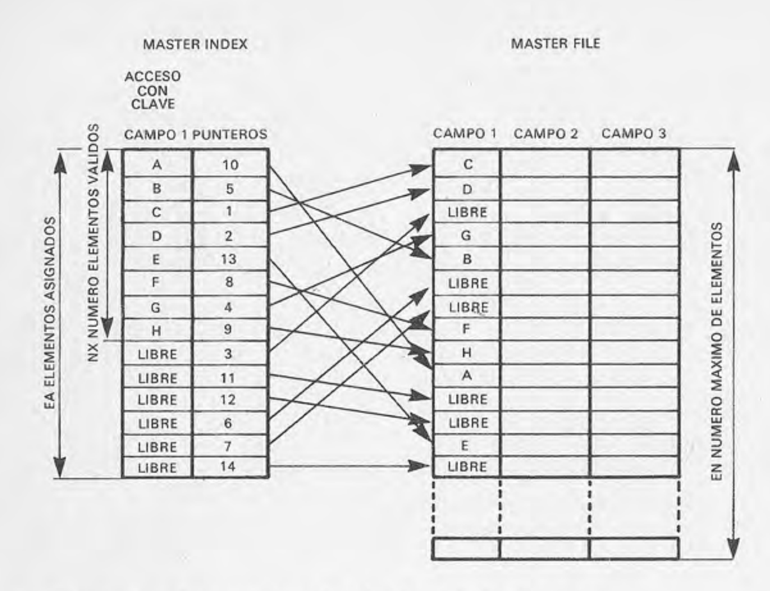


Figura 4.—Acceso con el método del diccionario.

Los dos ficheros están físicamente separados aunque, desde el punto de vista lógico, constituyen un archivo único e indivisible, teniendo que acceder siempre al diccionario para conseguir de forma ordenada las informaciones del archivo Base. De este modo sólo es posible el acceso mediante clave, es decir, por medio del primer campo del registro Base, supuesto como identificador prioritario.

Nada nos impide construir posteriores **diccionarios clave-puntero,** en donde la clave pueda ser uno cualquiera de los campos del registro Base; consideramos que cuatro índices auxiliares pueden ser suficientes (figura 5).

Por supuesto, en una estructura multiacceso de este tipo resultan complejas y largas las fases de introducción y eliminación, teniendo no solamente que actualizar el registro único del MASTER FILE y reordenar el MASTER INDEX, sino efectuar también una operación análoga con todos los ficheros índice auxiliares. Para hacer más rápido el procedimiento, tendremos siempre actualizado el ficher MASTER INDEX y permitiremos la actualización de los índices auxiliares *solamente con un comando explícito.*

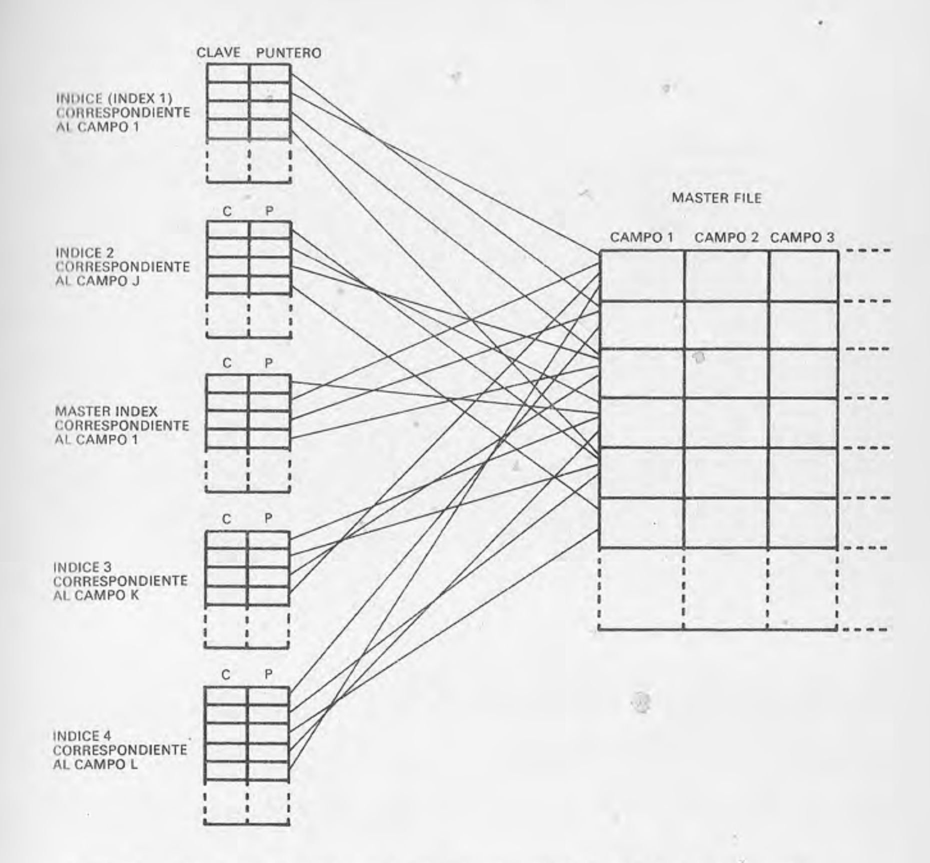


Figura 5.—Archivo multiacceso con el método del diccionario.

La estructura del archivo Base comienza a tomar consistencia y nos falta ahora un fichero auxiliar en donde almacenar el estado de la base de datos, es decir:

- El número de registros válidos NX.
- El número de elementos asignados EA, que es la suma de los elementos válidos y de los eliminados. La intercalación prevé ocupar el espacio que queda disponible a partir de los eliminaciones y solamente al agotarse este último asignar nuevos elementos.
- La existencia o no de un archivo controlado de listas, con su definición por un indicador de cadena FL (flag-link).

- El tipo de impresora conectada PT (0 para conexión en paralelo, 1 para conexión en serie).
- El número de columnas suceptibles de control por la impresona CN.
- El indicador de disco lleno DF.
- El número del campo asociado al primer índice auxiliar IA (1,0).
- La longitud de la clave (ocasionalmente truncada) correspondiente al primer índice auxiliar IA (1,1).
- La validez del primer índice auxiliar IA (1,2)

y otras informaciones análogas correspondientes a los tres últimos casos, pero relativas al índice auxiliar segundo, tercero y cuarto.

Son, pues, en total 18 los **parámetros** que permiten determinar de forma estable el estado del archivo y dichos parámetros se almacenarán en un fichero secuencial que denominaremos BASE DATOS CONTROL o, de forma abreviada BD.CTRL. En resumen, para la gestión de la base de datos tendremos:

BD.CTRL	:	control de la base de datos, fichero secuencial.
MST.INX	:	diccionario prioritario, fichero aleatorio.
MST.FL	:	archivo de datos, fichero aleatorio.
INDEX.1	:	primer diccionario auxiliar, fichero aleatorio.
INDEX.2	:	segundo diccionario auxiliar, fichero aleatorio.
INDEX.3	:	tercer diccionario auxiliar, fichero aleatorio.
INDEX.4	:	cuarto diccionario auxiliar, fichero aleatorio.

Una vez definida la estructura del archivo Base, vamos a considerar ahora la forma en que se puede conseguir una gestión de lista y cómo estructurarla para un control fácil y rápido.

A partir del estudio teórico anteriormente desarrollado se deduce que una gestión de lista exige punteros que conecten los registros entre sí y con la cabecera de la lista. En nuestro caso, la cabecera de la lista es lógicamente cada registro base individual y de aquí la necesidad de asignarles un puntero. No obstante, una solución de tal naturaleza obliga a asignar espacio en el fichero MASTER FILE, incluso en las aplicaciones en las que se quiera renunciar a la gestión de listas.

En este caso, se prefiere una solución alternativa que, además, permite hacer más rápida la gestión sucesiva. En vez de tener los punteros dirigidos a las listas en el interior de los registros podemos considerar su extracción y la elaboración de un fichero aleatorio (que denominaremos LINK ENTRY o LNK.ENTRY), constituido por un vector cuyos elementos son las cabeceras de las listas. Para comprobar la existencia (en el archivo Listas) de registros en correlación con el i-ésimo registro Base, bastará leer el i-ésimo

elemento del vector LINK ENTRY; si este último es 0 no están conectados los registros y, en caso contrario, el número constituirá el puntero al primer registro de la lista asociada.

Una lista **monodireccional** está constituida por pares registro-puntero en donde este último indica el par sucesivo en correlación; en esta ocasión preferimos subdividir dicho acoplamiento lógico en un vector de punteros (fichero LINK POINTERS o LNK.PTR) y un fichero aleatorio de registro, LINK FILE o LNK.FL, que contiene las informaciones adicionales del registro Base de pertenencia. Resumimos este en la figura 6.

No se trata de una gestión muy sencilla, lo admitimos, pero volveremos más adelante sobre esta materia.

Nos falta todavía una serie de parámetros que permiten conocer la estructura del registro correspondiente al fichero LINK, es decir, el número de campos y para cada uno de ellos:

- Nombre del campo.
- Tipo del campo (1 alfanumérico, 2 numérico).
- Longitud del campo.

Este conjunto de parámetros constituirá un fichero de control del LINK (cadena) que denominaremos LNK.CTRL.

Resumiendo, el archivo de listas estará constituido por los ficheros siguientes:

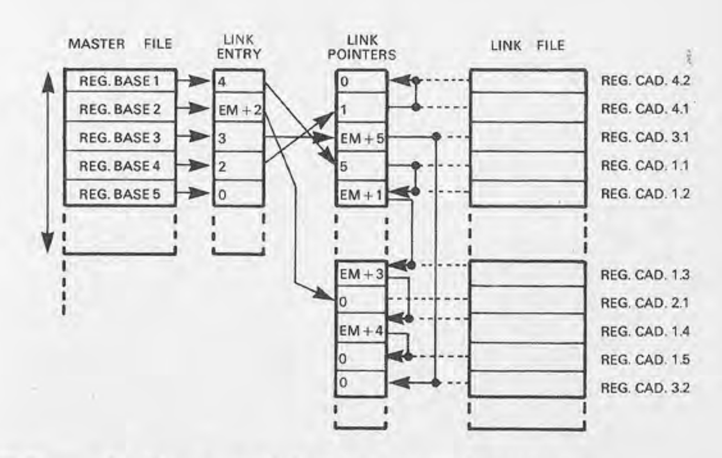


Figura 6.—Estructura del archivo controlado de listas.

LNK.CTRL : fichero secuencial que contiene los parámetros.
LNK.ENTRY : fichero aleatorio que contiene el vector de acceso a las listas (cabeceras).
LNK.PTR : fichero aleatorio que contiene el vector del puntero de gestión de lista.
LNK.FL : fichero aleatorio que contiene los registros controlados de lista.

Hemos definido la estructura del archivo Base y del archivo controlado de listas, pero nos falta todavía la definición de algunos parámetros tales como:

- El número máximo de registros base admitidos.
- El número de los campos del registro base y para cada campo:
  - — El nombre del campo.
  - — El tipo del campo (1 alfanumérico, 2 numérico).
  - — La longitud del campo.

Además, podría ser interesante "parametrizar" también la posición y organización física de los ficheros, almacenando para cada uno:

- El nombre del fichero.
- La longitud del registro (0 para secuencial).
- La ranura ("slot") de pertenencia.
- La unidad de disco de pertenencia

elaborado luego de forma automática las correspondientes cadenas de comando DOS.

Introduciremos dicho conjunto de parámetros en un posterior fichero secuencial (SISTEMA CONTROL o SIST.CTRL). Este fichero residirá siempre en la ranura 6, unidad 1, y será el *único fichero fijo,* a partir del cual se puede definir cualquier configuración y asignación física de la base de datos.

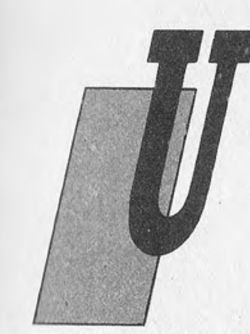
Para optimizar el programa definiremos para cada comando OPEN, READ, WRITE, CLOSE unos vectores O$(), R$(), W$() y C$(), para poder ejecutar el comando en el fichero deseado de forma directa. Por ejemplo, PRINT D$; O$(I), en donde I es el índice que sirve para indicar el fichero correspondiente, según la clave mostrada a continuación. Es cierto que la "parametrización" hace menos legible el listado, pero se obtiene una notable versatilidad y sencillez de gestión.

	SIST.CTRL	Control del sistema de gestión completo (secuencial).
0	MST.INX	Indice maestro o diccionario principal (aleatorio).
1	MST.FL	Archivo base (aleatorio).
2	BD.CTRL	Control de la base de datos (secuencial).
3	LNK.CTRL	Control del archivo controlado de listas.
4	LNK.ENTRY	Vector de conexión del archivo base a las listas (de carácter aleatorio).
5	LNK.PTR	Vector de los punteros para gestión de las listas (de carácter aleatorio).
6	LNK.FL	Archivo controlado de listas (de carácter aleatorio).
7	INDEX.1	Primer índice auxiliar.
8	INDEX.2	Segundo índice auxiliar.
9	INDEX.3	Tercer índice auxiliar.
10	INDEX.4	Cuarto índice auxiliar.

Figura 7.—Ficheros que constituyen la estructura de la base de datos.

CAPITULO IV

EL PROGRAMA



na vez definida la estructura no queda más que proceder a un análisis "top-down" de arriba a abajo del programa, que subdividiremos en dos partes bien distintas:

- Configuración del sistema, o SETUP.
- Gestión de BASE DATOS, o BD.

La primera permitirá configurar el sistema, es decir, definir el formato del archivo y, en el caso del MASTER FILE, a cuántos campos constituyen el registro y cuáles son los nombres, tipos y longitudes a ellos asociados. Se trata de definir los parámetros contenidos en el SIST.CTRL sin, por supuesto, cargarlos a mano. Lo mismo puede decirse con respecto a los ficheros BD.CTRL y LNK.CRTL.

De este modo se puede **personalizar** la base de datos que, aparte de las funciones estándar, debe poder archivar direcciones, catálogos o tablas sin tener que tocar ninguna sentencia del programa.

La segunda parte del programa permitirá, por el contrario, operar sobre el archivo preestablecido en las fases de:

- Introducción de datos.
- Borrado.
- Búsqueda y modificación.
- Listado.
- Cambio de índice de búsqueda.

Las dos partes anteriores constituirán programas bien distintos, pero mutuamente llamables, puesto que condensarlos en un solo programa podría ser "crítico" para la memoria de 48K del Apple.

La figura 1 representa cómo se controlará la **base de datos.** Una vez editados los dos avances faltarán los ficheros de control y por ello, al iniciarse el programa sin existir el fichero SIST.CTRL, se pasará el control al programa de CONFIGURACION DEL SISTEMA (SETUP), que permitirá configurar el sistema de gestión y, posteriormente, proceder a una nueva iniciación satisfactoria de la base de datos.

Las limitaciones de espacio para esta sección no nos permiten desarrollar el programa de configuración del sistema (SETUP) y la gestión de la base de datos (BD) y, al tener que elegir, decidimos dedicarnos al programa de configuración del sistema, de modo que podamos encontrarnos en los siguientes capítulos en condiciones para cargar el programa de gestión, después de haber asimilado el flujo y el listado del programa de configuración del sistema que nos disponemos a explicarles.

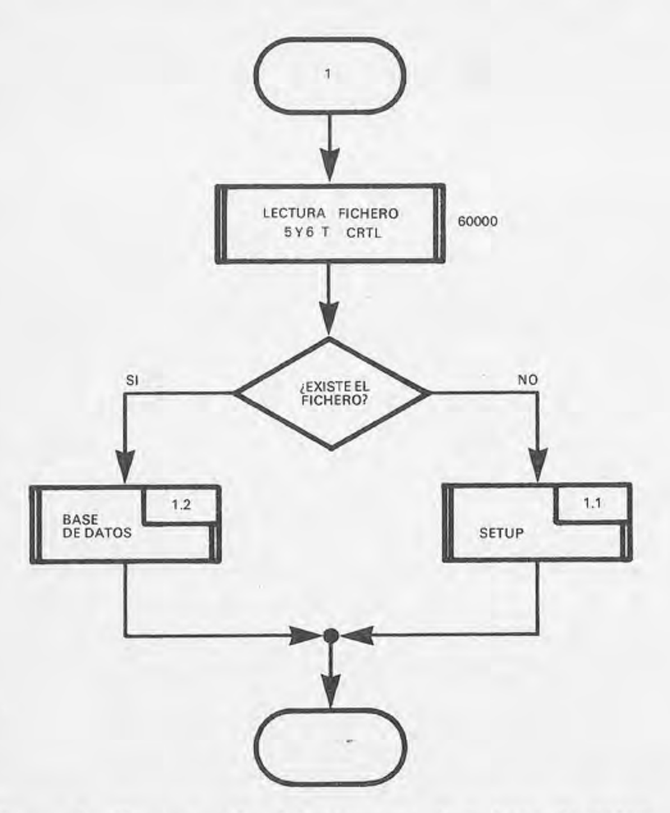


Figura 1.—Organización del programa de la base de datos.

Este modo de proceder está un poco en contradicción con el método de exposición de "arriba-abajo"; en efecto, una vez definida la estructura de los datos, y al tratarse de dos programas bien definidos cuyas interacciones se realizan solamente con la estructura antes aludida, consideramos que es más agradable para el lector concentrarse en un desarrollo del programa de configuración del sistema llegando hasta la codificación para volver luego a la parte superior en los siguientes capítulos, con plena dedicación a la gestión de la estructura ya existente.

Antes de continuar, es preciso que el lector tenga conocimiento del hardware necesario: basta disponer de un ordenador Apple de 48K y, como mínimo, un disco flexible en la ranura 6 de la unidad 3. Por supuesto, si se quiere tener unos archivos algo voluminosos, con el riesgos de saturar el único disco, sería necesario pasar al nivel de 2 o 3 unidades.

Definición de la base de datos

El programa de configuración del sistema (SETUP), siguiendo el diagrama de flujo de la figura 2, se inicia en la línea 60000 con la llamada a una subrutina (62000) de inicialización de las variables y con la lectura de dos ficheros del sistema, STRL.CTRL y BD.CTRL. En las inicializaciones aparecen dos variables, RF y RS, que contienen la cantidad de memoria libre en el programa BD y en el programa SETUP, respectivamente; por supuesto, en el caso de que se hagan modificaciones en los programas, será necesario actualizar dichos valores.

La primera vez que se inicia el programa no existirán estos ficheros y por ello en lugar de proseguir en secuencia y llegar al menú (60500) se activa ONERR GOTO 60050 que inicia la creación de la base de datos. Esto no ocurrirá en todas las demás ocasiones en que se ejecute el programa SETUP, puesto que se llegará directamente al menú que controla la posibilidad de una nueva configuración parcial del sistema o de su ampliación.

En la figura 3 podemos ver de qué modo se controla esta creación inicial: una vez que hemos entrado en la rutina de error se nos pregunta si el error verificado (cuyo código se encuentra en la dirección 222) es el que se esperaba, es decir END OF DATA (FIN DE DATOS), que es el mensaje emitido por el sistema operativo de disco (DOS) cuando se trata de leer un fichero no exis-

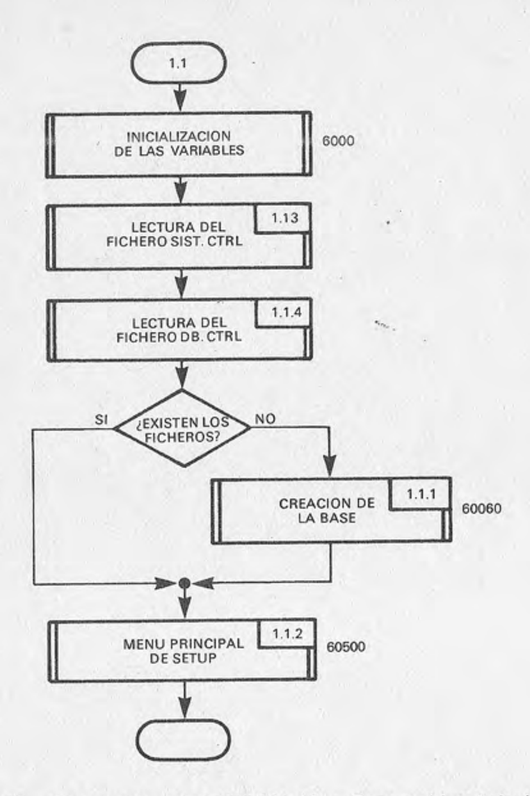


Figura 2.—Estructura del SETUP (Configuración del sistema.)

tente. En caso afirmativo, se prosigue desactivando ONERR (por cuanto que ya no es necesaria) mediante la instrucción POKE 216,0.

Después de haber definido en dónde asignar la base de datos (1.1.1.1) y cuáles son los campos requeridos para cada registro base (1.1.1.2), se pueden definir la longitud efectiva del registro R(0) y R(1) del INDICE y de la base de datos (línea 60095). Podemos escribir ahora el fichero SIST.CTRL (1.1.1.3) y proceder a su nueva lectura (1.1.3) para tener a nuestra disposición las cadenas de control que crea la rutina de lectura, procediendo a escribir el fichero BD.CTRL (1.1.1.4) y, finalmente, preparar en disco el espacio para los ficheros MASTER INDEX (MST.INX) y MASTER FILE (MST.FL) (1.1.1.5).

Después de esta asignación de espacio en disco, para cuya operación se puede tardar algunos minutos, se presenta la situa-

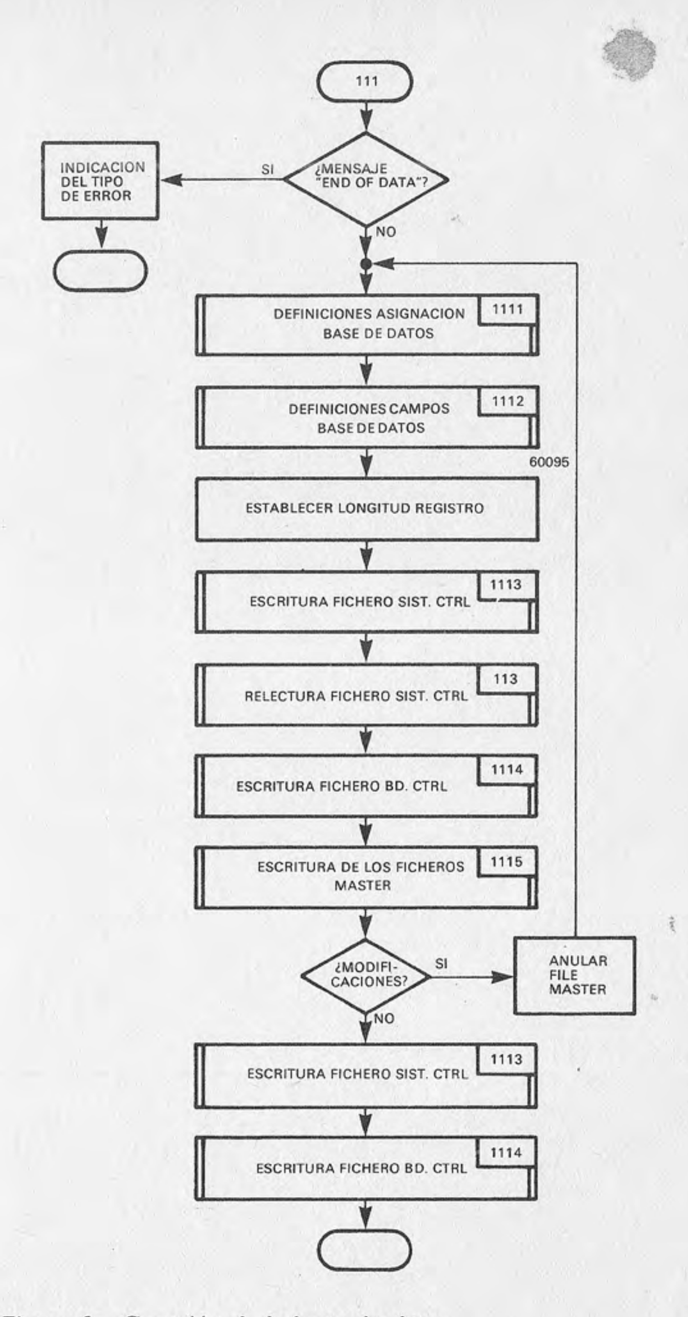


Figura 3.—Creación de la base de datos.

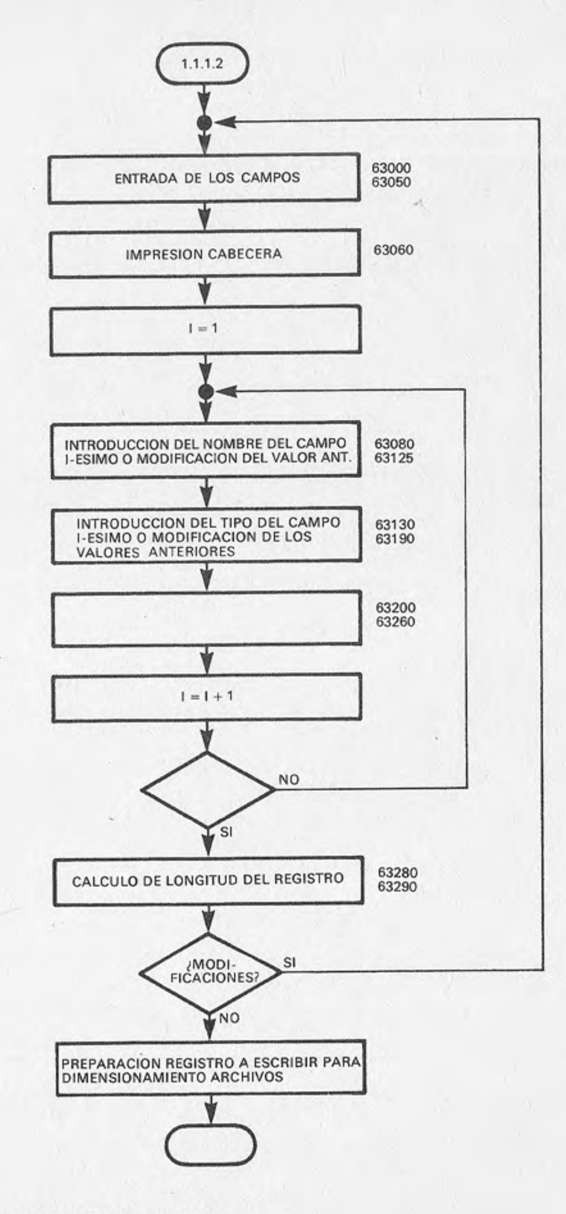
ción, es decir el número máximo de registros suceptibles de control: si la configuración no satisface, podemos modificarla todavía (en efecto, se puede volver a empezar anulando los ficheros maestros y volviendo a la definición de la asignación de la base de datos). Por el contrario, si todo es correcto, es escriben los ficheros SIST.CTRL (1.1.1.3) y BD.CTRL (1.1.1.4) definitivos.

En la figura 4 vemos cómo se produce la definición de la asignación de la base de datos: ante todo leyendo los datos (instruc-




```
62500 REM PRIMERA VEZ
62510 HOME:CN=80:RESTORE
62520 FOR I=0 TO 10:READ A$:NEXT
62530 FOR I=0 TO 10:READ R(I):NEXT
62540 FOR I=0 TO 10:READ S(I):NEXT
62550 FOR I=0 TO 10:READ D(I):NEXT
62555 FOR I=0 TO 9:READ E(I):NEXT
62560 VATB 9:HTAB 9:INPUT "BASE DE DATOS EN UNIDAD:";A$:A=VAL(A$)
62570 IF A<>1 AND A<>2 THEN 62560
62580 PRINT :HATB 22:INPUT "SEGURO? ";A$
62590 IF LEFT$(A$,1)<>"S" THEN 62560
62600 FOR I=0 TO2:D(I)=A:NEXT
```

Figura 4.—Definiciones de la asignación de la base de datos: flujo y listado.



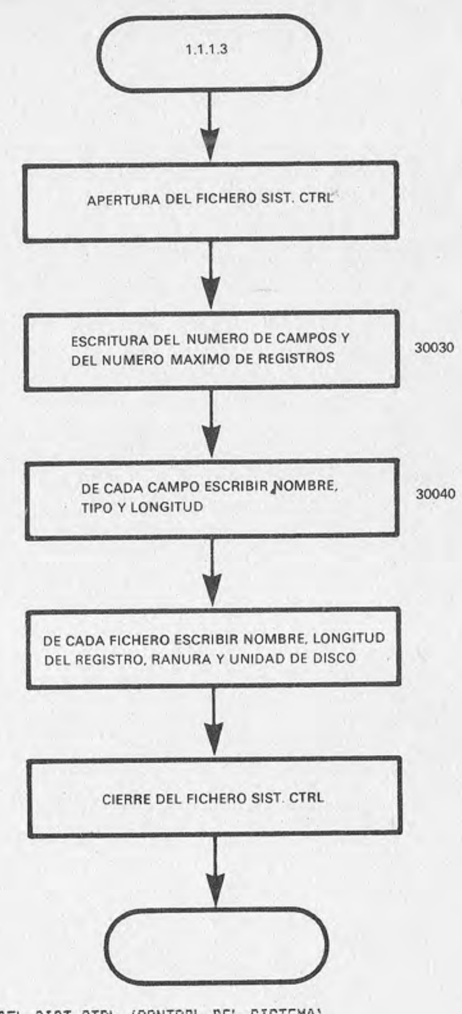

```
63000 REM CONFIGURACION DEL SISTEMA
63020 HOME:HTAB 18:PRINT "CONFIGURACION DEL SISTEMA"
63030 HTAB 13:PRINT "CUANTOS CAMPOS?: ";:IF NC THEN PRINT NC;
63040 IF NC>NM OR NC<1 THEN 63020
63050 VATB 2:HTAB 27:CALL -868:PRINT NC
```

```
63060 VATB 4:HTAB 11:PRINT "NOMBRE";:HTAB 21:PRINT "TIPO (A/N)";:HTAB 25:PRINT "
      LONGITUD"
63070 FOR I=1 TO NC
63080 VATB I+5:CALL -868:IF I>10 THEN HTAB 2
63090 PRINT I;"CAMPO";
63100 HTAB 10:IF LB$(I)<>"" THEN PRINT LB$(I);
63110 HTAB 10:INPUT "";A$
63113 IF A$="" AND LB$(I)="" THEN 63080
63116 IF A$<>"" THEN LB$(I)=A$
63120 VTAB I+5:HTAB 10:PRINT LB$(I)
63140 IF TP(I)=1 THEN INPUT "ALFANUMERICO ";A$
63145 IF TP(I)=2 THEN INPUT "  NUMERICO ";A$
63150 IF TP(I)=0 THEN INPUT " ";A$
63160 VTAB I+5:HTAB 20
63170 IF A$="A" THEN TP(I)=1:CALL -868:PRINT "ALFANUMERICO":GOTO 63200
63180 IF A$="N" THEN TP(I)=2:CALL -868:PRINT "   NUMERICO ":GOTO 63200
63190 IF A$<>"" OR TP(I)=0 THEN 63130
63200 VTAB I+5:HTAB 35:CALL -868
63210 IF LL(I)<>0 THEN PRINT LEFT$(T$,4-LEN(STR$(LL(I))));LL(I);:HTAB 35
63220 INPUT "";A$
63230 IF A$="" AND NOT LL(I) THEN 63200
63240 IF A$<>"" THEN LL(I)=INT (VAL(LEFT$(A$,3)))
63250 IF LL(I)<1 OR LL(I)>29 THEN 63200
63260 VTAB I+5:HTAB 35:CALL -868:PRINT LEFT$(T$,4-LEN(STR$(LL(I))));LL(I)
63270 NEXT
63280 LR=NC:FOR I=1 TO NC:LR=LR+LL(I):NEXT
63290 VTAB 21:PRINT "LONGITUD REGISTRO "LR" BYTES"
63300 VTAB 23:HTAB 20:INPUT "MODIFICAR? ";A$
63310 IF LEFT$(A$,1)<>"N" THEN 63000
63320 ST$=""
63330 J=LR-1:IFJ>255 THEN J=255
63340 FOR I=1 TO J:ST$=ST$+FR$:NEXT
63350 RETURN
```

Figura 5.—Definición de los campos: flujo y listado.

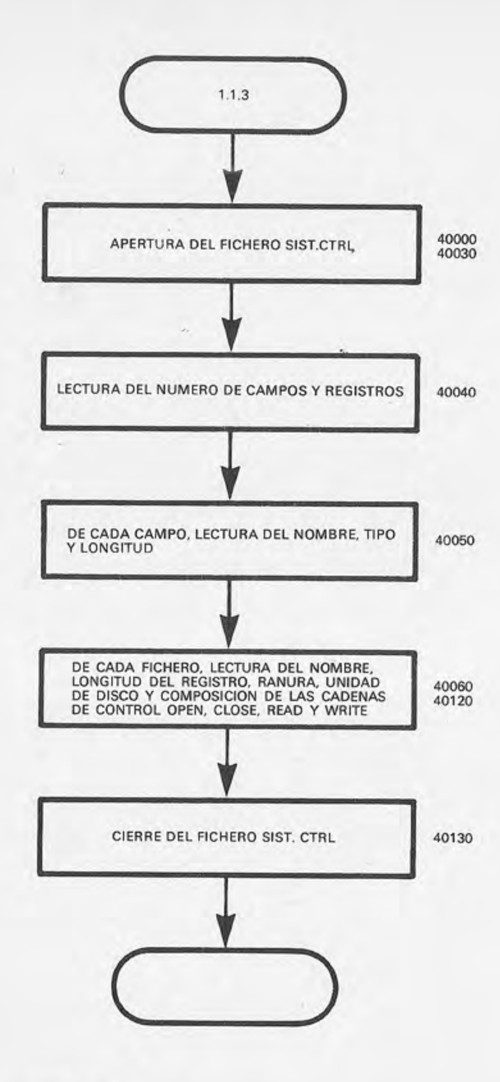
ciones DATA que aparecen el final del listado) se asumen los parámetros (nombre del fichero, longitud de registro, ranura, unidad) prefijados (líneas 62520-62550) y a continuación se pregunta en qué unidad se quiere ubicar el archivo principal de la base de datos y se procede a la sustitución del valor anterior por el actual.

Más compleja es la definición de los campos en el interior del registro (figura 5): la rutina está estructurada de modo que se pueda, con las mismas instrucciones, definir un campo o modificar el valor anterior del mismo, prosiguiendo con su ejecución hasta que




```
30000 REM ESCRITURA DEL SIST.CTRL (CONTROL DEL SISTEMA).
30010 PRINT D$;O1$"SIST.CTRL,S6,D1"
30020 PRINT D$;W1$"SIST.CTRL"
30030 PRINT NC","EM
30035 FOR I=0 TO 9:PRINT E(I):NEXT
30040 FOR I=1 TO NC:PRINT LB$(I)","TP(I)","LL(I):NEXT
30050 RESTORE
30060 FOR I=0 TO 10:READ A$:PRINT A$","R(I)","S(I)","D(I):NEXT
30070 PRINT D$;C1$"SIST.CTRL"
30080 RETURN
```

Figura 6.—Escritura del fichero SIST.CTRL: flujo y listado.




```
40000 REM LECTURA DEL FICHERO SIST.CTRL
40010 B$(0)="":B$(1)=",R"
40020 PRINT D$;1$"SIST.CTRL,S6,D1"
40030 PRINT D$;R1$"SIST.CTRL"
40040 INPUT NC,EM
40045 FOR I=0 TO 9:INPUT E(I):NEXT
40050 FOR I=1 TO NC:INPUT LB$(I),TP(I),LL(I):NEXT
40060 FOR I=0 TO 10
40070 L$="":INPUT A$,R(I),S(I),D(I):IF R(I) THEN L$=",L"+STR$(R(I))
40080 D$(I)=01$+A$+L$+",S"+STR$(S(I))+",D"+STR$(D(I))
```



```
40090 R$(I)=R1$+A$+B$(R(I)>0)
40100 W$(I)=W1$+A$+B$(R(I)>0)
40110 C$(I)=C1$+A$
40120 NEXT
40130 PRINT D$;C1$"SIST.CTRL"
40140 RETURN
```


Figura 7.—Lectura del fichero SIST.CTRL: flujo y listado.

no se obtenga una situación satisfactoria. Es importante destacar que esta definición, una vez confirmada, ya no será modificable a menos que se destruya todo el archivo; en efecto, en esta parte del programa, como se vio con anterioridad, se entra si, y solamente si, no existen los ficheros de sistema, es decir, la primera vez que se inicia el programa.

De cada campo se define:

- Nombre: con una longitud máxima de 9 caracteres.
- Tipo: alfunumérico o numérico.
- Longitud: comprendida entre 1 y 29.

Para los campos de tipo **alfanumérico** se comprobará solamente la longitud, mientras que para los de tipo **numérico** se comprobará también si son efectivamente números. Al comienzo de la rutina se pide el número de campos que se quiere constituyan el registro base; el límite máximo de este valor lo hemos establecido en 14, pero puede aumentarse a voluntad, con la simple preocupación de que luego, en el programa BD, se controle de forma adecuada la paginación de los datos en la pantalla.

En el cálculo de la longitud del registro, se precisa tener presente que dentro de cada campo, en el interior del registro, existe el carácter de retorno de carro (CR) separador y que, por consiguiente, la longitud en bytes viene dada por la suma de las longitudes de los campos más el número de campos (un retorno de carro cada uno). Al final de la rutina se prepara una cadena de caracteres a escribir en el disco para asignar el espacio necesario para el archivo. Esta cadena, constituida por caracteres no utilizables de inmediato, tiene la misma longitud que el registro o, si este último tiene una longitud superior a 255 bytes, tendrá una longitud de 255 bytes, siendo esta dimensión suficiente para asignar todo el espacio necesario.

En la rutina se tiene una instrucción CALL-868; se trata de la llamada a una función del sistema que borra de la pantalla la línea actual, desde la posición del cursor hasta el final de la línea.

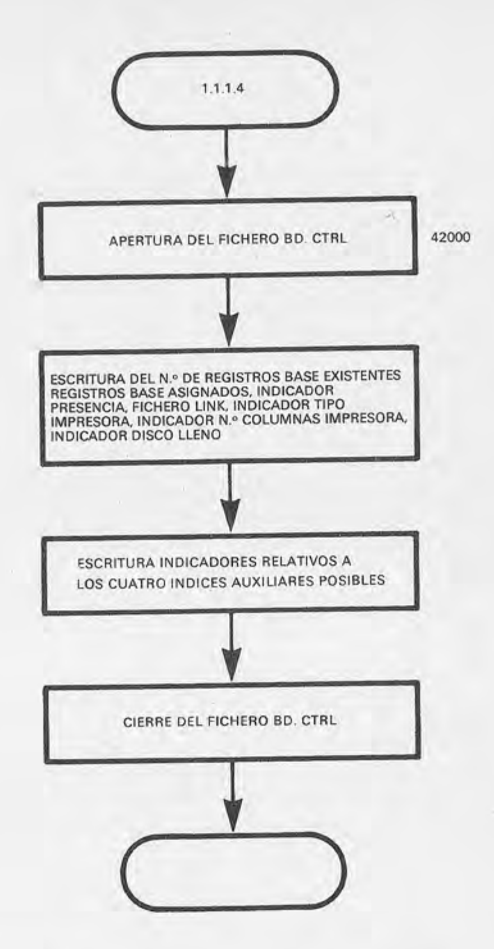
En este punto podemos ver la estructura de los ficheros del sistema SIST.CTRL y BD.CTRL, que contienen los datos de descripción del ambiente en el que operan los dos programas de configuración del sistema y de la base de datos.

En las figuras 6 y 7 vemos la escritura y lectura del fichero SIST.CTRL; en las figuras 8 y 9 podemos observar la escritura y lectura del fichero BD.CTRL. En todo el programa se constata que la escritura del fichero SIST.CTRL va inmediatamente seguida por la rutina de lectura, por cuanto que esta última, además de leer los parámetros de diversos ficheros, construye las cadenas 0$(I), R$(I), W$(I), C$(I), para el acceso a los ficheros; por consiguiente, cada vez que se varía uno de los parámetros de un fichero (por ejemplo, el correspondiente a la ranura), es preciso volver a ejecutar esta operación para poder tener el acceso correspondiente.

Prosiguiendo con un estudio más profundo del programa, vemos en el diagrama de flujo de la figura 11 como se asigna el espacio en disco para los ficheros maestros. Inicialmente, se pregunta cuántos registros se quieren que constituyan el archivo base; en este punto, el programa comienza a escribir en el disco, al mismo tiempo en INDEX y en MASTER. En el primero se escribe una clave ficticia y el número de orden y en el segundo la cadena preparada en la fase de definición de los campos.

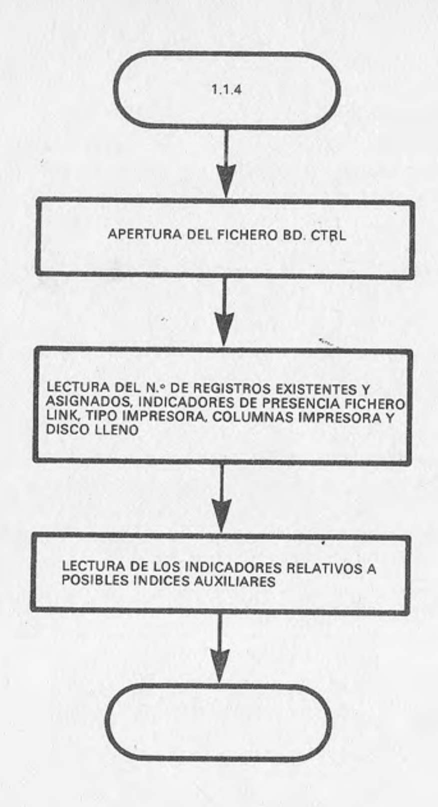
Existen dos posibilidades: que en el disco elegido haya espacio suficiente para contener el archivo o que no. En esta segunda hipótesis se activa ONERR GOTO 60360 que está contenida en la línea 60160 y partiendo del último registro que se ha tratado de escribir y procediendo hacia atrás se verifica cuál es el último par "index-master" escrito de forma correcta (ello porque el sistema operativo DOS escribe físicamente en el disco no para cada comando WRITE, sino solamente cuando ha llenado un buffer de 256 bytes; y por consiguiente, a priori no se sabe en qué momento se verifica el error del disco lleno). En ambos casos, al llegar a la línea 60280, en "I" está contenido el número máximo de registros base.

Se calcula ahora el número de claves suceptibles de contenerse en la memoria del programa BD, para que se mantengan en ella con el fin de tener un acceso más rápido. Como el número máximo de registros utilizables por la base de datos en la configuración elegida, se elige el mínimo entre el número de registros que son suceptibles de almacenamiento en disco y el número de claves contenida en la memoria. En este punto, como se vió en el diagrama de flujo de la figura 3, también es posible hacer modificaciones volviendo a comenzar desde el principio; por ejemplo, variando la longitud de la clave o de algunos campos, para aumentar el número de informaciones suceptibles de control.

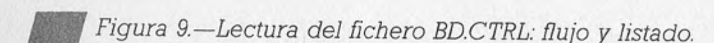



```
42000 REM ESCRITURA DEL FICHERO BD.CTRL (CONTROL BASE DE DATOS).
42010 PRINT D$;D$(2)
42020 PRINT D$;W$(2)
42030 PRINT NX","EA","FL","PT","CN","DF
42040 FOR I=1 TO 4
42050 PRINT IA(I,0)","IA(I,1)",IA(I,2)
42060 NEXT
42070 PRINT D$;C$(2)
42080 RETURN
```


Figura 8.—Escritura del fichero BD.CTRL: flujo y listado.




```
35000 REM LECTURA FICHERO BD.CTRL
35010 PRINT D$;D$(2)
35020 PRINT D$;R$(2)
35030 INPUT IA(I,0),IA(I,1)IA(I,2)
35040 FOR I=1 TO 4
35060 NEXT
35070 PRINT D$;C$(2)
35080 RETURN
```


Figura 9.—Lectura del fichero BD.CTRL: flujo y listado.


```
15700 REM ASIGNACION DE FICHEROS
15710 VTAB 20:INPUT "CORRECTAS RANURA Y UNIDAD? ";A$
15720 FI A$="S" OR A$="" THEN RETURN
15730 VATB 20:CALL -958:PRINT "RANURA "S(P1);
15740 HATB 6:INPUT "";A$:S(P1)=VAL(A$)
15750 VATB 20:HTAB 21:PRINT "UNIDAD";D(P1);
15760 HATB 27:INPUT "";A$:D(P1)=VAL(A$):PRINT
15780 HATB 23:INPUT "MODIFICACIONES? ";A$
15790 IF A$="S" THEN 15730
15800 RETURN
```

Figura 10.—Listado de la subrutina de asignación de ficheros.

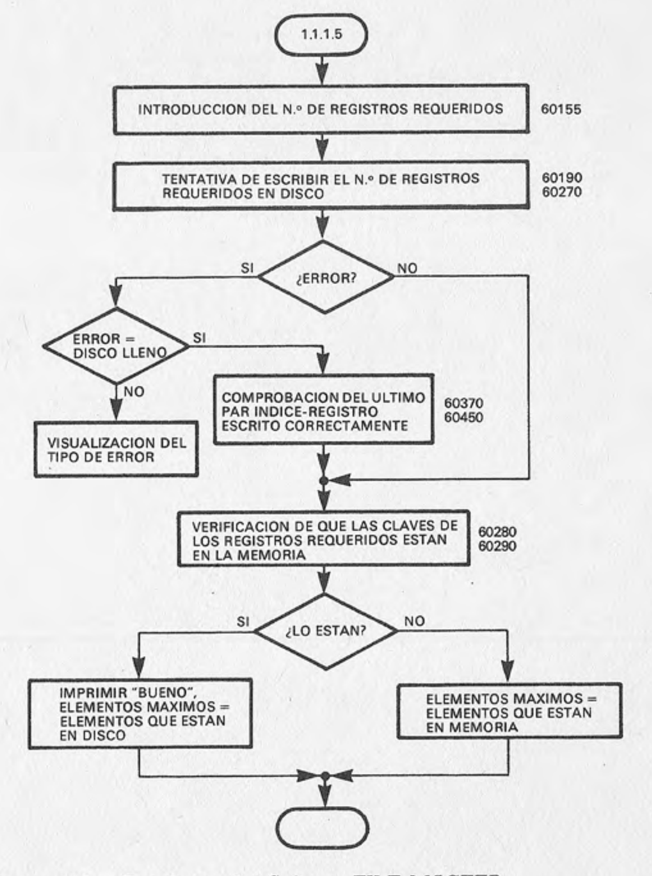


Figura 11.—Escritura del fichero FILE MASTER.

CAPITULO V

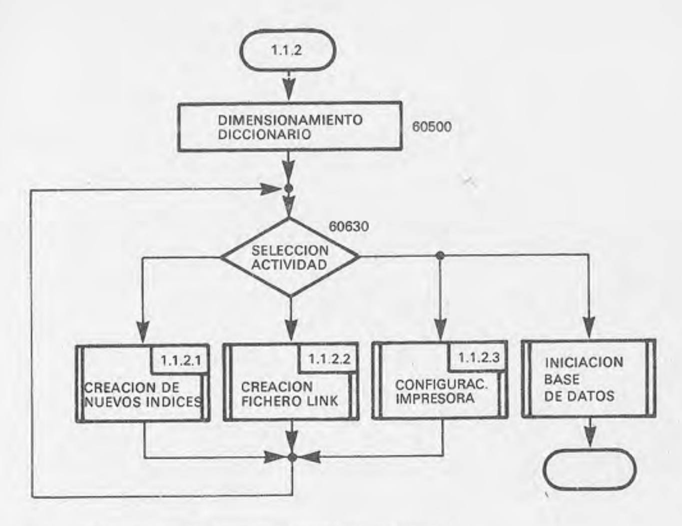
EL MENU PRINCIPAL



Acabamos de definir la estructura inicial de la base de datos y, por consiguiente, el programa llega al menú principal, como llegará todas las veces que desde BD se llama a este programa para reconfigurar parte del sistema. Las variables usadas son:

VARIABLE	DESCRIPCION
A	Variable numérica auxiliar utilizada a nivel local.
A$	Variable auxiliar utilizada mucho para la entrada.
B$	Variable auxiliar utilizada a nivel local.
B$(I)	Vector auxiliar.
C$(I)	Cierre del fichero i-ésimo.
CI$	Cadena que contiene el valor 'close'.
CN	Número máximo de columnas de la impresora.
D$	Variable de cadena con carácter de control ctrl-d(CH$(1)).
D(I)	Unidad de pertenencia del fichero i-ésimo.
DF	Indicador de disco lleno (si es 1 indica disco lleno).
EA	Indica el número de registros base asignados (comprende también los anulados).

FL	Indicador LINK (si 1 indica la existencia de lista).
FR$	Variable que contiene CHR$(95), es decir, registro libre.
I	Indice auxiliar.
IA$	Cadena auxiliar para algoritmo de ordenación.
IA%	Variable numérica auxiliar para algoritmo de ordenación.
IA(I,J)	Parámetros índice i-ésimo (J=0 número campo; 1 longitud clave; 2 validez).
IX$(I)	Clave registro i-ésimo.
IX%(I)	Dirección registro i-ésimo.
J	Indice auxiliar.
L	Indice.
LB$(I)	Nombre campo i-ésimo (número máximo de caracteres=8).
LL(I)	Longitud campo i-ésimo (máximo 29 caracteres).
LR	Longitud registro.
NC	Número de campos.
NM	Número máximo de campos.
NX	Número de elementos válidos del archivo base.
O$(I)	Apertura fichero i-ésimo.
OI$	Cadena que contiene el valor 'open'.
PI	Indice de fichero (parámetro para rutina de asignación).
PT	Tipo interface impresora (0-en paralelo, 1-en serie).
R	Indice auxiliar para ordenación.
R$(I)	Lectura fichero i-ésimo.
R(I)	Parámetro que indica longitud registro fichero i-ésimo (0-secuencial).
RI$	Cadena que contiene el valor 'read'.
FR	Memoria libre para claves en el programa BD.
RS	Memoria libre para claves en el programa SETUP.
S(I)	Parámetro que indica ranura de pertenencia del fichero i-ésimo.
SI$	Cadena de CHR$(95), utilizada para asignación de espacios en disco.
T$	Cadena que contiene 28 espacios en blanco.
TP$(I)	Tipo campo i-ésimo (1-alfanumérico, 2-numérico, 0-no definido).
W$(I)	Escritura fichero i-ésimo.
WIS	Cadena que contiene el valor 'write'



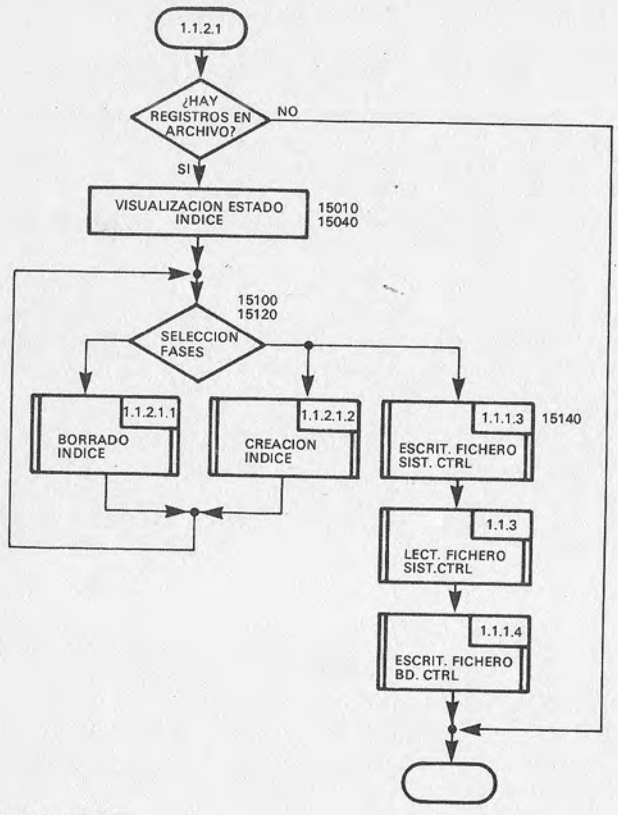

```
60500 REM MENU PRINCIPAL PROGRAMA CONFIGURACION SISTEMA
60505 DIM IX$(EM),I%(EM),PI%(EM)
60510 HOME:HTAB 10:PRINT "RECONFIGURACION SISTEMAS"
60520 VTAB 5
60530 PRINT " 1- ACTUALIZACION INDICES ADICIONALES"
60540 PRINT :PRINT " 2- CREACION ARCHIVOS CONCATENADOS"
60550 PRINT :PRINT " 3- CONFIGURACION IMPRESORA"
60560 PRINT :PRINT " 4- RETORNO AL MENU PRINCIPAL"
60570 VTAB 22:HTAB 20:PRINT "B.B.I."
60600 VTAB 15:HTAB 10:INPUT "CUAL? ";A$
60610 I=VAL(A$)
60620 IF I<1 OR I>4 THEN 60510
60630 ON GOSUB 15000,20000,25000,61000
60640 GOTO 60510
```

Figura 1.—Menú del programa de configuración: flujo y listado.

A partir de la figura 1 vemos que las actividades posibles en este punto son:

- La gestión de nuevos índices de acceso (1.1.2.1) además del principal, siempre presente y actualizado.
- La creación de los ficheros concatenados (1.1.2.2).
- La definición de la impresora conectada (1.1.2.3).
- Salida de este programa con la iniciación del programa BD.

Comenzamos a partir de la gestión de los índices auxiliares (figura 2): si todavía no se grabaron los registros, no será posible



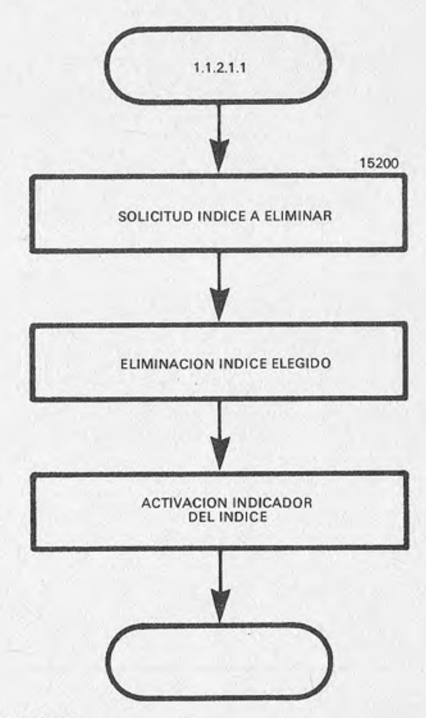

```
15000 REM GESTION INDICES
15005 IF NOT NX THEN RETURN
15010 HOME:HTAB 15:PRINT "INDICES EXISTENTES":PRINT
15020 FDOR I=1 TO 4
15030 PRINT I"   "LB$(IA(I,0);:HATB 15:PRINT "S"S(I+6)" D"D(I+6);:IF NOT IA(I,2)
     THEN HTAB 22:PRINT "NO";
15040 PRINT "  ACTUALIZADO":NEXT
15060 VTAB 8:PRINT " 1-ELIMINACION"
15070 PRINT " 2-CREACION"
15080 PRINT " 3-FIN"
15100 PRINT : INPUT "CUAL? "; A$: A% = VAL(A$)
15110 IF A$<1 OR A%>2 THEN 15000
15120 ON A% GOSUB 15200,15300
15130 IF A%<>3 THEN 15000
15140 GOSUB 30000:GOSUB 40000:GOSUB 42000
15150 RETURN
```


Figura 2.—Gestión de índices auxiliares: flujo y listado.

crear ningún índice y por ello la fase terminará de inmediato. De no ser así se presenta la situación actual, visualizando, para cada uno de los cuatro índices posibles, el nombre del campo al que se hace referencia, la ranura y la unidad en la que está almacenado y si está actualizado o no.

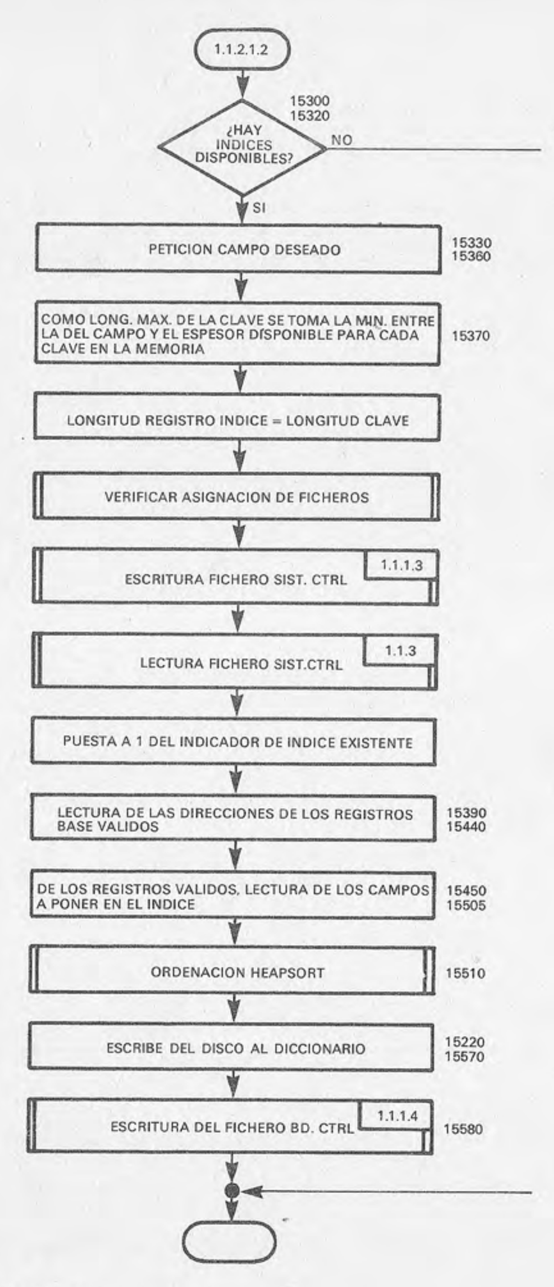
Las elecciones posibles son tres: eliminar un índice que ya no es necesario (1.1.2.1.1.), creación de uno nuevo (1.1.2.1.2.) o terminar las operaciones registrando la nueva operación en los ficheros de sistema.

Para **borrar** un índice (figura 3) basta con ejecutar un comando DELETE <nombre fichero> y actualizar el indicador IA(i, j) co-




```
15200 REM ELIMINACION INDICE
15210 PRINT :INPUT "CUAL ELIMINAR? ";A$:A=VAL(A$)
15220 IF A<1 OR A>4 THEN 15200
15230 IF NOT IA(A,0) THEN RETURN
15250 PRINT D$"DELETE ";MI$(D$(A+6),6,7);:RIGHT$(D$(A+6),6)
15260 IA(A,0)=0:IA(A,2)=0
15270 RETURN
```


Figura 3.—Eliminación de un índice: flujo y listado.




```
15300 REM CREACION INDICE
15310 FOR P=1 TO 4:IF IA(P,0) THEN NEXT
```



```
15320 IF P>4 THEN  RETURN
15330 VTAB 14:INPUT "QUE CAMPO? ";A$:A=VAL(A$):CALL -958
15340 IF <2 OR A> NC OR TP(A)=2 THEN 15330
15350 VTAB 14:HTAB 18:PRINT LB$(A);HTAB 29:INPUT "LO CONFIRMA? ";A$
15355 IF LEFT$(A$,1)<> "S" THEN 15330
15360 IA(P,0)=A:VT(0)=A
15365 GOSUB 16000:PRINT :IF NOT OK THEN 15330
15370 IA(P,1)=INT((FRE(0)-2000)/EM):IF IA(P,1)>LL(A) THEN IA(P,1)=LL(A)
15375 R(P+6)=IA(P,1)+6:P1=P+6:PRINT P"   ";:GOSUB 15700:GOSUB 30000:GOSUB 40000
15380 IA(P,2)=1
15385 VTAB 20:CALL -958:HTAB 13:FLASH:PRINT "LECTURA CAMPOS":NORMAL
15390 PRINT D$;O$(0):PRINT D$;O$(1)
15400 FOR J=1 TO NX
15410 PRINT D$;R$(0);J
15420 INPUT A$:INPUT IX%(J)
15470 PRINT D$;R$(1);IX%(J)
15480 FOR K=1 TO A
15490 INPUT A$
15500 NEXT
15505 IX$(J)=LEFT$(A$,IA(P,1)):NEXT
15507 VTAB 20:CALL -958:HTAB 5:FLASH:PRINT "ATENCION! ORDENACION EN CURSO":NORMA
L
15510 GOSUB 500
15515 VTAB 20:CALL -958:HTAB 12:FLASH:PRINT "ESCRITURA INDICE":NORMAL
15518 ONERR GOTO 17000
15520 PRINT D$;O$(P+6)
15525 PRINT D$;W$(P+6);0:PRINT NX
15530 FOR J=1 TO NX
15540 PRINT D$;W$(P+6);J
15550 PRINT IX$(PI%(6)):PRINT IX%(PI%(J))
15560 NEXT
15570 PRINT D$;C1$
15575 POKE 216,0
15580 GOSUB 42000
15590 RETURN
```


Figura 4.—Creación de un índice: flujo y listado.

rrespondiente. En la línea 15250 se ve que el nombre del fichero se toma a partir de la cadena O$ correspondiente al índice en cuestión y de la misma cadena se toman las indicaciones de la ranura y de la unidad de pertenencia.

Para **crear** un nuevo índice (figura 4) es necesario, ante todo, que no estén los cuatro ocupados.

Después de haber solicitado el número del campo que se quiere utilizar se calcula la longitud máxima de la clave conteni-

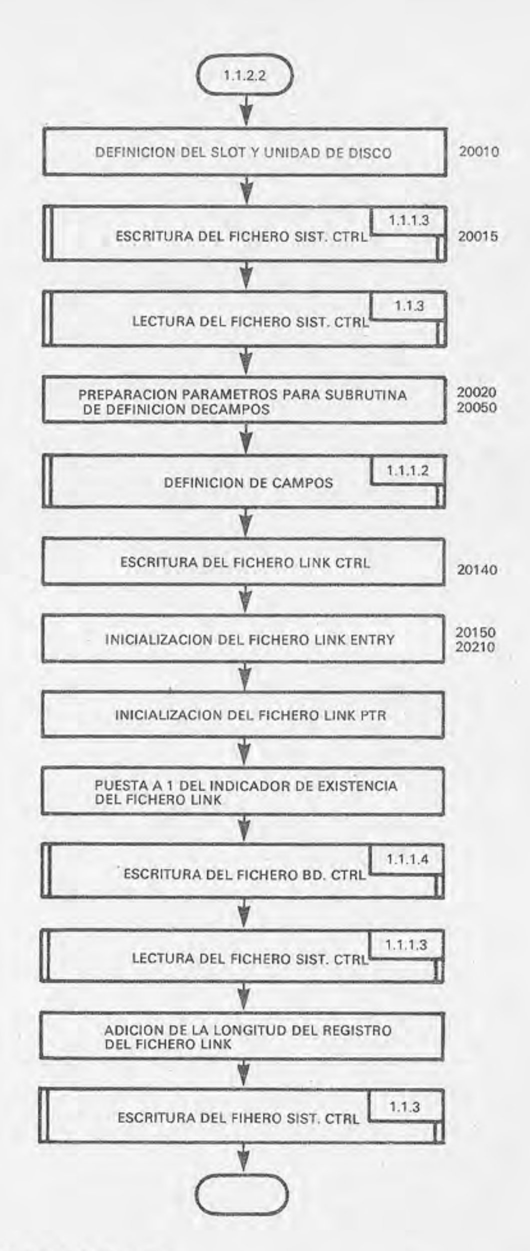
ble en la memoria del programa SETUP (línea 15370) y, puesto que las claves deben estar contenidas en memoria para su ordenación, será preciso cerciorarse de que el espacio es suficiente para todas ellas. En la línea 15375 se valora la valoración del registro índice como dimensión de la clave +6 bytes que contienen la dirección del registro base correspondiente a cada clave y los separadores(CR) entre clave y dirección.

Inmediatamente después se ejecuta una subrutina (que comienza en la línea 15700) que permite variar la ranura ("slot") y la unidad en la que se almacena el índice. Corresponde al operador tener cuidado en no introducir los índices en un disco que no tenga el espacio libre necesario para contenerlos; de no ser así, durante la escritura del índice se generará el mensaje de error "DISK FULL" (en cualquier caso, siempre podrá volverlo a crear en otro disco). Al escribir y volver a leer el fichero SIST.CTRL se dispondrá de las cadenas de control actualizadas para el acceso a los datos.

Para crear el índice solicitado será necesario, en este punto, leer a partir del MASTER INDEX cuáles son los registros del archivo válidos, almacenando las direcciones correspondientes en el vector IX%(líneas 15390 y 15440; se lee también la clave principal en la variable A$ pero, al no ser necesaria, no se almacena). Al saber ahora cuáles son los registros válidos del MASTER FILE podemos leerlos uno a uno y almacenar en el vector IX$ el campo correspondiente al índice objeto de creación, desechando los otros campos. Al final de esta operación, tendremos en los vectores IX$ e IX% todas las informaciones necesarias para tener acceso a los registros según otro índice y no nos queda más que ordenarlos mediante la rutina **heapsort** para poder efectuar la búsqueda binaria, almacenarlos en el disco elegido y registrar en el fichero BD.CTRL la existencia de un nuevo índice actualizado.

El diagrama de flujo de la figura 5 representa la fase de creación de los ficheros LINK, que comienza con la definición de qué ranura (slot) y unidad de disco se quiere utilizar para estos datos; luego se crean las cadenas de control con el procedimiento habitual de escritura y relectura del fichero SIST.CTRL.

Para definir los campos que se quieren que constituyan el registro concatenado se utiliza el mismo procedimiento empleado para el registro base, variando simplemente algunos parámetros y utilizando las mismas variables (por lo que después de ello será necesario volver a leer el fichero SIST.CTRL para restablecer la situación anterior). La definición de los campos crea parámetros similares a los del registro principal que se escriben en el fichero LNK.CTRL. La inicialización de los otros ficheros necesarios para la gestión de los ficheros concatenados consiste simplemente en poner a 0 todo el fichero LINK ENTRY, teniendo en cuenta que no



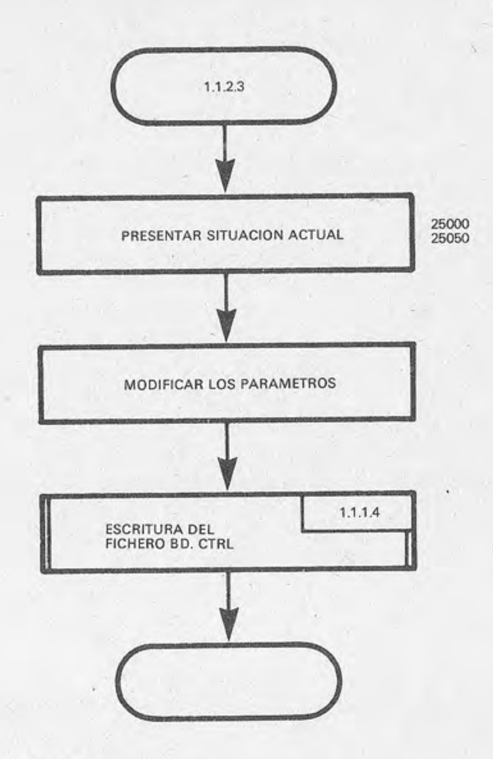

```
20000 REM CREACION NUEVO FICHERO
20010 HOME:VTAB 9:PRINT "RANURA"S(3)" UNIDAD"D(3):P1=3:GOSUB 15700:FOR I=4 TO 6:
S(I)=S(3):D(I)=D(3):NEXT
20015 GOSUB 30000:GOSUB 40000
```

```
20020 NC=0:NM=14
20030 FOR I=1 TO NM
20040 LB$(I)="":TP(I)=0:LL(I)=0
20050 NEXT
20060 GOSUB 63000
20070 REM ESCRITURA FICHERO LINK.CTRL(CONTROL CADENA)
20075 VATB 23:CALL -958:FLASH:HTAB 13:PRINT "PREPARACION":NORMAL
20080 PRINT D$;O$(3)
20090 PRINT D$;W$(3)
20100 PRINT NC
20110 FOR I=1 TO NC
20120 PRINT LB$(I)","TP(I)","LL(I)
20130 NEXT
20140 PRINT D$;C$(3)
20150 REM INICIALZACION FICHERO ENTRY
20160 PRINT D$;O$(4)
20170 FOR I=0 TO EM
20180 PRINT D$;W$(4);I
20190 PRINT 0
20200 NEXT
20210 PRINT D$;C$(4)
20220 PRINT D$;O$(5)
20230 PRINT D$;W$(5);0
20240 PRINT 1
20250 PRINT D$;C$(5);0
20260 FL=1:DF=0
20270 GOSUB 42000
20280 GOSUB 40000
20285 R(6)=LR:GOSUB 30000
20290 RETURN
```


Figura 5.—Creación del fichero LINK: flujo y listado.

existen listas (en realidad, solamente las hemos definido) y que debemos escribir en el fichero LINK. PTR cuál es el primer registro libre, es decir, el número 1. La conclusión de la fase es, como de costumbre, la escritura de los ficheros del sistema para actualizar la situación.

La última fase (figura 6) se refiere a la definición del tipo de **impresora** que se quiere utilizar. La diferencia está en el tipo de interface (que debe estar en el slot número 1). Existen dos posibilidades: en paralelo o en serie. El otro parámetro es el número máximo de caracteres que la impresora controla por cada línea (normalmente 80 ó 132). Estos parámetros también se registran en el fichero BD.CTRL.




```
25000 REM PARAMETROS IMPRESION
25010 HOME:HATB 12:PRINT "IMPRESORA TIPO:"
25020 VATB 5:HATB 5:IF NOT PT THEN PRINT "EN PARALELO";:GOTO 25030
25025 PRINT "EN SERIE";
25030 HTAB 20:PRINT "DE "CN " COLUMNAS"
25040 VTAB(15): HTAB(20): INPUT "MODIFICACIONES? ": A$: IF LEFT$(A$,1) <> "S"
THEN 25100
25050 VTAB 18:INPUT "IMPRESORA (S/P)? ":PT=0
25060 IF A$="S" THEN PT=1
25070 INPUT "MUMERO COLUMNAS? ";A$:CN=INT (VAL(A$))
25080 GOTO 25000
25100 GOSUB 42000
25110 HOME
```


Figura 6.—Configuración de la impresora: flujo y listado.

Hemos visto cómo crear la estructura de la base de datos, los ficheros concatenados y los índices auxiliares, cuáles son los parámetros necesarios para el control de todo el conjunto y cómo

configurar el sistema según las propias exigencias y las posibilidades de la máquina (por ejemplo, el número de unidades de que se dispone).

En el siguiente capítulo proseguiremos con el desarrollo, analizando el programa propiamente dicho de gestión de datos.

CAPITULO VI

GESTION DE DATOS

Llegamos por fin al "corazón" de la gestión de datos, cuya estructura se describió en los capítulos anteriores, esperamos que con buenos resultados. Los afortunados lectores que hayan tenido la posibilidad de teclear la fase de configuración del sistema (SETUP) no estarán precisamente muy entusiasmados puesto que, al tratarse de una fase preparatoria, no proporciona resultados tangibles inmediatos. Pedimos ahora un poco de paciencia y les aseguramos que, al finalizar, estarán en condiciones de realizar la gestión de la base de datos, en esta ocasión con resultados prácticos. Quien piense que así habrá puesto término al desarrollo del proyecto está en un error puesto que, como se indicó con anterioridad, nos proponemos concluir con una fase de listado, es decir, de visualización e impresión, muy sofisticada que ponga de manifiesto las ventajas de poder disponer de una base de datos bien estructurada.

La conclusión comprenderá también algún ejemplo que ponga a prueba su imaginación en diversos campos de aplicación que sean de su interés. Concluiremos, pues, la odisea de la base de datos en el siguiente capítulo, pero no terminará para el lector, que tratará de modificar, optimizar y "personalizar" su gestión.

Antes de comenzar, les recomendamos añadir inmediatamente la línea 62150 del programa SETUP

```
62150 RF=16000
```

puesto que con la introducción de la fase de listado quedará menos espacio de memoria para las variables de cadena y dicha mo-

dificación sirve de precaución contra posibles desbordamientos futuros de la capacidad de memoria.

Por si no está clara la estructura de datos controlados por listas trataremos con más detalle los conceptos correspondientes.

Cómo realizamos la gestión de los registros concatenados

A primera vista, la metodología que hemos elegido para controlar los registros concatenados puede parecer tediosa e incomprensible; no obstante, aunque este sistema hay que reconocer que alarga algo los tiempos de acceso presenta la gran ventaja de que solamente existe una conexión lógica, y no física, entre los registros base y los registros en cadena. Ello permite tener un archivo y "N" discos separados de registros concatenados que tienen también un formato diferente y poder introducir y utilizar, uno a uno, estos discos.

Veamos ahora con detalle cómo se manipulan las listas. Cuando el programa SETUP crea los ficheros LINK, genera en el disco elegido, y en el fichero LNK.CTRL, una descripción de los campos del registro LINK. Además crea tres ficheros: LNK.ENTRY, LNK.PTR y LNK.FILE.

LNK.ENTRY contiene tantos registros como registros base máximos hay en el archivo. En este fichero, cada registro es el puntero a la cabecera de la lista conectada al registro base correspondiente. El registro 1 de LNK.ENTRY contendrá, pues, el puntero a la cabecera de la lista unida al registro base número 1 y así sucesivamente. En el momento de la creación, este fichero se pone a cero en su totalidad, lo que significa que no hay cadenas unidas.

LNK.FILE contiene todos los registros concatenados. No necesita inicialización por cuanto que son precisamente los dos ficheros los encargados de establecer si un registro está lleno o vacío.

LNK.PTR contiene todos los punteros necesarios para la exploración y creación de las listas; esta es la razón por la que se hizo la elección de tener los punteros de las cadenas en un fichero separado de los datos propiamente dichos.

Veamos con un ejemplo cómo funcionan estos tres ficheros: teniendo el registro base número 3 conectado a una cadena de tres registros. Entonces, según puede verse en la figura 1, en el registro número 3 de LNK.ENTRY estará el primer registro en cadena en el fichero LNK.FILE (por ejemplo el 5) y en el registro número 5 de LNK.PTR estará el número del siguiente registro en cadena (por ejemplo, el 10); en el registro número 10 de LNK.FILE están los datos correspondientes, en el registro número 10 de

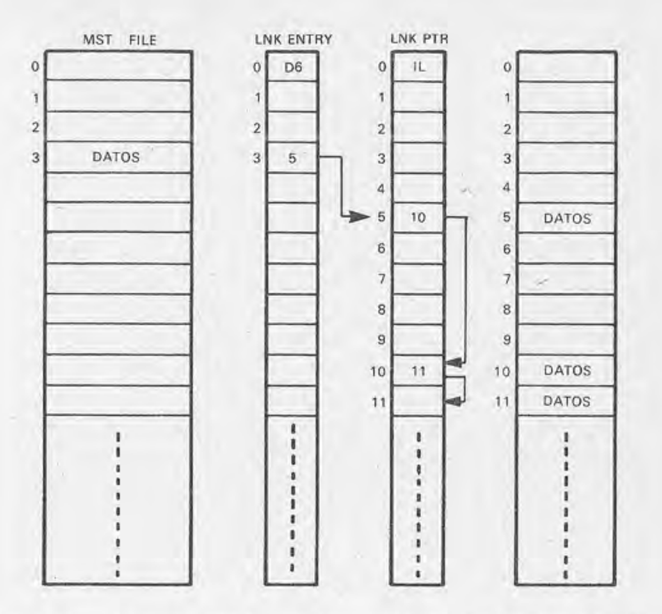


Figura 1.—Ejemplo del funcionamiento de los ficheros LINK.

LNK.PTR está el puntero del siguiente, y así sucesivamente hasta encontrar un cero en LNK.PTR, lo que indicará el final de la cadena.

Una atención especial merecen los registros de los ficheros LNK.ENTRY y LNK.PTR. Desde el momento en que se presignaron los ficheros con la definición previa de la extensión (con el fin de permitir almacenar en el mismo disco también otros datos, por ejemplo, los índices auxiliares) es necesario saber cuál será el siguiente registro libre y cuándo se hará la inserción de un nuevo registro, lo mismo que es necesario controlar los registros borrados, es decir, eliminados de una lista. Entonces, en el registro 0 del fichero LNK.PTR se mantiene el número de orden del primer registro libre en LNK.FILE; en realidad, en la inicialización se escribe 1, en este fichero, lo que representa el primer registro libre.

En lo que respecta a los registros borrados de listas preexistentes para una reutilización sucesiva se conectan en una lista de elementos libres y en el registro 0 del fichero LNK.ENTRY se escribe la cabecera de esta lista. Cuando se tenga que hacer una inserción, si existe una lista libre, se extrae un registro de ella y se reutiliza, actualizando la cabecera de la cola. Si no existiera, entonces se ve cuál es el siguiente registro libre, actualizando luego

A	Variable numérica auxiliar
A$	Variable de cadena auxiliar
A%	Variable entera auxiliar
B$	Variable de cadena auxiliar
B$(*)	Vector de cadena utilizado en la lectura del fichero SIS.CTRL
C	Variable numérica empleada en el borrado
C$(*)	Vector que contiene las cadenas de control 'CLOSE' de los ficheros
C1$	Cadena 'CLOSE'
CA	Parámetro para rutina de búsqueda, que indica llamada de borrado
CI$	Valor ASCII correspondiente a control -I
CL	Columnas de formato de lista
CM	Parámetro para la rutina de visualización campos
CN	Longitud máxima de la línea de impresión
CR$	Valor ASCII correspondiente a 'RETURN'
D$	Valor ASCII correspondiente a control-b
DF	Indicador de disco lleno
DL	Puntero a la cabecera de la lista libre
E$	Contiene la cadena 'RETURN para'
EA	Número de elementos de BD asignados
EC$	Contiene el valor ASCII 'esc'
EL	Puntero al último registro a listar
EM	Número máximo de registro base
EY$(*)	Vector de cadena de apoyo para introducción y borrado
EY%(*)	Vector de enteros de apoyo para introducción y borrado
F	Indicador de encontrado, o no, al término de la subrutina de búsqueda
FD	Indicador de borrado de cadena
FI	Indicador de índice de acceso activo
FL	Indicador de registro modificado
FR$	Carácter indicador de clave libre
I	Variable de utilidad
I%(*)	Matriz índices de validez para la fase de listado
IA(**)	Matriz que contiene los parámetros de los índices auxiliares
IL	Siguiente registro libre en el fichero LINK
IV	Número de elementos válidos en el índice auxiliar
IX$(*)	Vector de las claves principales
IX%(*)	Vector de los punteros correspondientes a las claves principales
J	Variable de utilidad
K	Contador
K0$	Clave principal cuando se utilizan los índices auxiliares
KM$	Clave de búsqueda
L	Puntero para búsqueda binaria
L$	Utilizado en la creación de las cadenas de control DOS
L%(*,*)	Matriz longitudinal de campos en listado BD y LINK
LB$(*,*)	Matriz nombre de los campo BD y LINK
LK	Longitud clave para búsqueda binaria
LL(*,*)	Matriz longitudinal de los campos BD y LINK
LP	Número de líneas del módulo de impresión
M$	Cadena 'modificaciones?'
MA$(*)	Matriz de las claves máximas de selección para listado BD y LINK
MI$(*,*)	Matriz de las claves mínimas de selección para listado BD y LINK
N	Puntero o número registros base actual
NC(*)	Número campos BD y LINK
NM	Número máximo de campos
NP	Indicador nueva página
NU	Número último registro válido del índice principal después de la inserción o fusión
NX	Número de registros base existentes
O$(*)	Vector de cadenas de control DOS 'OPEN' de los ficheros
O%(*,*)	Matriz de los campos en totalización horizontal, utilizada en el listado
O1$	Cadenas 'OPEN'
P	Parámetro de salida de la búsqueda binaria, que apunta al registro encontrado
PL	Indicador de presencia LINK
PT	Tipo de impresora (Ø-en paralelo, 1-en serie)
R$(*)	Vector de las cadenas de control 'READ' de los ficheros
R(*)	Vector de longitudes de registros
R1$	Cadena 'READ'
RA	En exploración de cadena es el puntero al registro actual
RO	En exploración cadena es el puntero al registro anterior
RB$(*,*)	'Matriz de cadena de apoyo'
TV$(*,*)	Matriz que contiene el registro base o LINK en gestión
S	Indicador de impresión activa
S(*)	Vector indicador de las ranuras («slot») de pertenencia de los ficheros
SE	Indicador del resultado de la prueba de selección
SL	Puntero al primer elemento de listado
Sd(*)	Vector indicador de la presencia de selección para BD y LINK
T(*)	Tipo lista horizontal o vertical para BD y LINK
T1$	Cadena utilizada para visualizar la longitud de los campos
TP(*,*)	Matriz tipo de campo para BD y LINK
TT(*,*)	Matriz de las totalizaciones verticales
V%(*,*)	Matriz de los campos en totalización vertical
W$(*)	Vector que contiene las cadenas de control 'WRITE' de los ficheros
W1$	Cadena 'WRITE'
X	Indica el registro concatenado a escribir en el disco



Figura 2.—Lista completa de las variables de la subrutina de listado.

el indicador correspondiente. En el programa estos dos punteros son: DL (cabecera cola libre) e IL (siguiente registro libre).

Las **inserciones** se realizarán siempre en la cabecera de la lista, quedando una cola tipo "LIFO" ("último en llegar primero en salir"), para que la operación sea más rápida.

Análisis de los flujos

Antes de analizar los flujos, hemos de establecer una premisa y es la de que, como se puede ver, lamentablemente no están estructurados. Ello presenta la ventaja de la velocidad de ejecución y de la sencillez del programa.

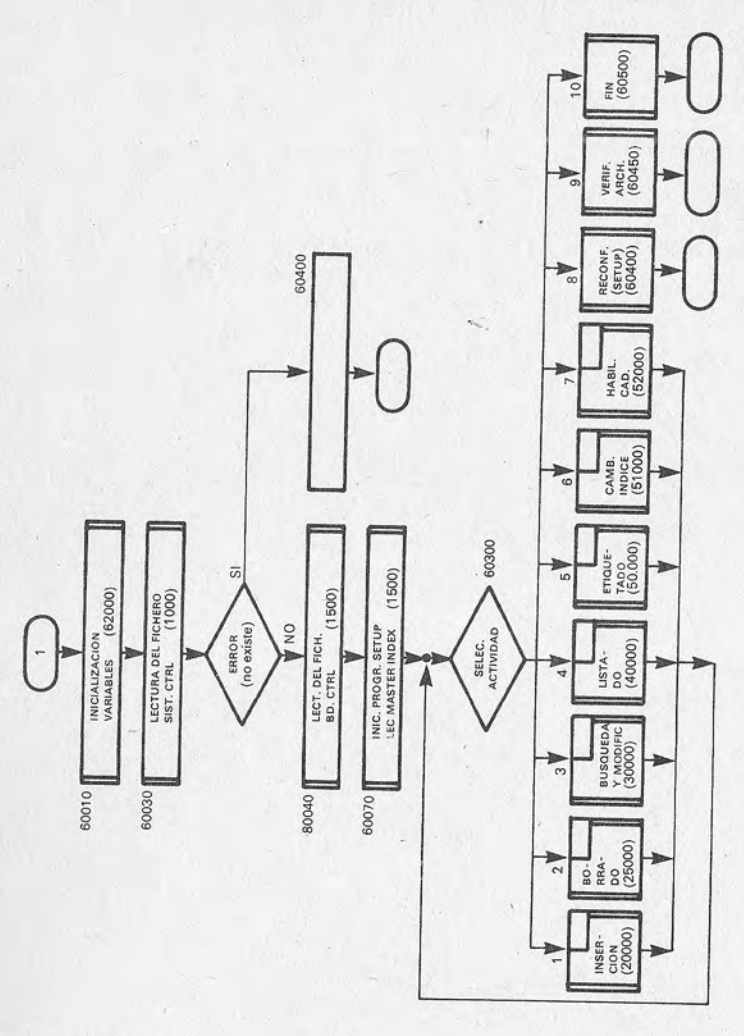
Comenzamos ahora (figura 3) a observar la estructura superior de la base de datos. Después de la ejecución de una rutina que inicializa las variables se lee el fichero SIST.CTRL y según la situación se tendrán dos formas diferentes de proseguir el proceso.

En la primera vez que se ejecuta el programa el fichero no existe y, por consiguiente, se activa ONERR GOTO que inicia el programa SETUP visto en el capítulo anterior, con el fin de configurar el sistema.

A continuación, será posible leer el fichero y continuar la ejecución del programa BD. Los datos contenidos en el fichero son:

- NC(0): número de campos contenidos en el registro base.
- EM: número máximo de registros archivo principal.
- LB$(I,0), TP(I,0), LL(I,0): para cada campo, su nombre, tipo y longitud.
- A$,R(I),S(I),D(I): para cada fichero, su nombre, longitud de registro, "slot" y unidad de disco.

La subrutina de lectura crea también, con los datos extraídos del fichero, las cadenas 0$, R$, W$ y C$, que contienen los comandos OPEN, READ, WRITE y CLOSE, respectivamente, de los ficheros que componen el sistema.




```
1 PRINT "MAXFILES 3"
10 GOTO 60000
100 REM ******************************
101 REM ***      A P P L E  I I      **
102 REM **                           **
104 REM ** BASE DE DATOS PERSONAL **
105 REM ******************************

700 REM LECTURA CAMPOS DE BASE DE DATOS
710 PRINT R$(XF);P:INPUT IX$:INPUT IX%
720 PRINT R$(1);IX%:J=0
```



```
800 REM LECTURA DE CAMPOS
810 FOR I=1 TO N(J):INPUT RV$(I,J):NEXT
820 PRINT D$
830 RETURN
900 REM NO ENCONTRADO
910 IF F THEN  RETURN
920 VTAB 3:HTAB 15:PRINT "NO HAY"
930 VTAB 8:HTAB 22:GOSUB 58000
940 POP:RETURN

10000 REM LECTURA DEL FICHERO LINK CTRL (CONTROL CADENA)
10010 PRINT O$(3)
10020 PRINT R$(3):INPUT NC(1)
10030 FOR I=1 TO NC(1)
10040 INPUT LB$(I,1),TP(I,1),LL(I,1)
10050 NEXT
10060 PRINT C$(3)
10070 FOR I=4 TO 6:PRINT O$(I):NEXT
10080 PRINT R$(4);0:INPUT DL
10090 PRINT R$(5);0:INPUT IL
10100 PRINT R$(5);INPUT IL
10110 PRINT D$
10120 RETURN

38000 REM DISCO LLENO
38010 IL=IL-1
38020 PRINT O$(5)
38030 PRINT W$(5);0:PRINT IL
38040 PRINT C$(5)
38100 HOME:HTAB 15:PRINT "DISCO LLENO":PRINT :GOSUB 58000
38140 DF=1 :GOSUB 2000
38150 POKE 216,0
38160 RUN
40000 REM LISTADO

58000 REM ESPERA TECLADO
58010 INPUT "PULSE RETURN ";A$
58020 RETURN
59000 REM OVERFLOW BD
59010 HOME:PRINT "NO HAY LUGAR PARA OTROS REGISTROS"
59020 GOSUB 58000
59030 RETURN
60000 REM PRUEBA PRIMERA VEZ (DEFINICIONES DE LA ASIGNACION DE LA BASE DE DATOS)
60010 GOSUB 62000
60020 ONERR GOTO 60400
60030 GOSUB 1000
60040 GOSUB 1500
60050 POKE 216,0
60070 GOSUB 10500
```

```
60100 REM MENU PRINCIPAL
60105 TEXT
60110 HOME:HTAB 11:PRINT "BASE DE DATOS PERSONAL"
60120 VTAB 3:PRINT "REGISTROS PRESENTES EN ARCHIVO";NX
60130 PRINT "MAXIMOS REGISTROS POSIBLES      ";EM
60140 VTAB 6:HTAB 10:PRINT "FASES DE ACTIVIDAD:"
60150 VTAB 8:PRINT " 1- INSERCION"
60160 PRINT " 2- ELIMINACION"
60170 PRINT " 3- BUSQUEDA Y/O MODIFICACION"
60180 PRINT " 4- LISTADO"
60190 PRINT " 5- ETIQUETA"
60200 PRINT " 6- CAMBIO DE INDICE"
60210 PRINT " 7-";:IF FL THEN PRINT "DES"
60215 PRINT " HABILITACION CADENAS"
60220 PRINT " 8- RECONFIGURACION DEL SISTEMA"
60230 PRINT " 9- VERIFICACION ARCHIVO"
60290 PRINT "10- FIN"
60295 VTAB 23:HTAB 10:PRINT "D & V PRODUCTION"
60300 VTAB 19:HTAB 10:PRINT "CUAL? ";A$:I=VAL(A$)
60320 IF (I>1 AND I<6) AND NOT NX THEN 60100
60330 ON I GOSUB 20000,25000,30000,40000,50000,51000,52000,60400,60450,60500
60340 SL(0)=0:SL(1)=0:GOTO 60100
60400 REM LANZAMIENTO PROGRAMA DE CONFIGURACION DEL SISTEMA
60410 PRINT D$"EJECUCION DE PROGRAMA DE CONFIGURACION DEL SISTEMA"
60450 PRINT D$"EJECUCION DEL PROGRAMA DE CHEQUEO"
60500 REM FINAL DEL PROGRAMA"
60510 PRINT C1$
60520 PRINT D$"LOCK LOGO,D1"
60530 PRINT D$"MAXFILES 3"
60540 HOME:PRINT "¡ ADIOS !"
60550 END
62000 REM INICIALIZACION VARIABLES
62010 I=0:T1$="-":FOR J=I TO 4:T1$=T1$+T1$:NEXT
62020 D$=CHR$(4):FR$=CHR$(95)
62030 NM=15
62040 DIM LB$(NM,1),TP(NM,1),LL(NM,1),RB$(NM,1),RV$(NM,1),EY$(25),EY%(25)
62050 O1$=D$+"OPEN":R1$=D$+"READ":W1$=D$+"WRITE":C1$=D$+"CLOSE"
62060 DIM MI$(NM,1),MA$(NM,1)
62070 FOR I=1 TO NM:FOR J=0 TO 1:MA$(I,J)=FR$:LB$(NM,J)="TOTAL":NEXT J,I
62080 DIM I%(NM,1),L%(NM,1),V%(NM,1),O%(NM,1),TT(NM,2),T(2),SL(1)
62090 CR$=CHR$(13):EC$=CHR$(27):CI$=(9)
62100 M$="MODIFICACIONES? "
62200 RETURN

1000 REM LECTURA DEL FICHERO SIST.CTRL
1010 B$(0)="":B$(1)=",R"
1020 PRINT O1$"SIST.CTRL,S6,D1"
1030 PRINT R1$"SIST.CTRL": INPUT NC(0),EM
```



```
1040 FOR I=0 TO 9:INPUT E(I):NEXT :LT=E(0)
1050 FOR I=1 TO NC(0):INPUT LB$(I,0),TP(I,0),LL(I,0):NEXT
1060 FOR I=0 TO 10
1070 L="":INPUT A$,R(I),S(I),D(I):IF R(I) THEN L$= ",L"=STR$(R(I))
1080 O$(I)=O1$+A$+L$+",S"+STR$(S(I))+",D"+STR$(D(I))
1090 R$(I)=R1$+A$+B$+(R(I))0)
1100 W$(I)=W1$+A$+B$(R(I))0)
1110 C$(I)=C1$+A$
1120 NEXT
1130 PRINT C1$"SIST.CRL"
1140 RETURN
1500 REM LECTURA DEL FICHERO BD.CTRL
1510 PRINT O$(2)
1520 PRINT R$(2):NX,EA,PL,PT,CN,DF
1530 FOR I=1 TO 4:INPUT IA(I,0),IA(I,1),IA(I,2):NEXT
1570 PRINT C$(2)
1580 RETURN
2000 REM ESCRITURA DEL FICHERO BD.CTRL
2010 PRINT O$(2)
2020 PRINT W$(2):PRINT NX","EA","PL","PT","CN","DF
2040 FOR I=1 TO 4:PRINT IA(I,0)","IA(I,1)","IA(I,2):NEXT
2070 PRINT C$ (2)
2080 RETURN

10500 REM LECTURA INDICES
10550 PRINT O$(0)
10560 PRINT O$(1)
10570 RETURN
```


Figura 3.—Menú del programa de Base de datos.

El programa prosigue con la lectura del fichero BD.CTRL, que contiene los datos actualizados sobre el estado del sistema y exactamente:

- NX: número de registros base existentes en el archivo.
- EA: número de registros base asignados (NX-número de registros utilizados ya pero borrados a continuación).
- PL: indicador de presencia de ficheros concatenados.
- PT: indicador del tipo de impresora.
- CN: número máximo de columna de impresión.
- DF: indicador de si el disco de los ficheros concatenados está lleno.

Y además, para cada uno de los cuatro índices auxiliares posibles: número de campo interesado, longitud en el índice (que puede ser menor que la longitud efectiva del campo) y validez, o no, del índice.

Conociendo el estado del sistema es posible leer ahora a partir del disco el índice principal, con el fin de tener en la memoria las claves de acceso, en el vector IX$, y las direcciones correspondientes, en el vector IX%.

Se añade, pues, al menú principal con lo que se permitirá la elección de las diversas actividades, a saber:

- INSERCION.
- BORRADO.
- BUSQUEDA Y MODIFICACION.
- LISTADO.
- ETIQUETAS (véase apéndice).
- CAMBIO DE INDICE.
- HABILITACION CADENAS.
- RECONFIGURACION DEL SISTEMA.
- VERIFICACION DE ARCHIVOS (véase apéndice).
- FINAL.

Inserción

Esta fase permite la inserción de los registros en el archivo base, comprobando la disponibilidad y efectuando la operación de manera rápida y segura. Si se quieren obtener mejores tiempos de búsqueda, los datos deberían estar estructurados de forma adecuada, lo cual está en contradición con el deseo de controlar la inserción de manera sencilla y rápida. Deberemos encontrar, pues, soluciones de compromiso.

La inserción de datos en la estructura quiere decir **añadir** un elemento o sustituir un elemento eliminado del archivo base (MST.FL) y efectuar la inserción ordenada en el fichero índice o MST.INX. La primera parte es muy sencilla, puesto que basta saber cuál es la posición libre.

La segunda podría resultar más difícil, puesto que, al tener que efectuar una ordenación o una inserción ordenada, nos veremos obligados, en el caso más desfavorable, a volver a escribir todo el fichero de las claves principales, operación que lleva un cierto tiempo que, no obstante, sería aproximadamente igual para una o varias inserciones. Por consiguiente, se tiene la exigencia de crear un vector disponible para el almacenamiento simultáneo

de las claves principales correspondientes a los nuevos elementos, que luego se utilizan para efectuar una inserción ordenada con el fichero de las claves (merge-fusión).

Dicho vector lógico está subdividido en dos vectores físicos: EY$ que contiene las claves y EY% que contiene los punteros correspondientes escritos en el fichero MST.FL. Dichos vectores se dimensionaron a 25 elementos para permitir un "bufffereado" de un máximo de 25 inserciones consecutivas, de las cuales, además, será necesario efectuar una fusión o intercalación (merge).

La secuencia de las instrucciones viene indicada en la figura 4. Como puede observar, se comprueba inmediatamente la posibilidad de insertar elementos y solamente en caso favorable se efectúan las acciones preparatorias, es decir, obsolencia de los índices auxiliares (que después de la inserción pueden actualizarse solamente a petición) y preparación del vector "entry" antes descrito con elementos de clave libre y puntero a los registros liberados por eliminaciones o nunca utilizados.

Una vez terminada la fase preparatoria el control pasa a la subrutina de gestión de entrada de campos con el parámetro CO=0 que indica la inserción. A continuación se efectúa un control de existencia de la clave de acceso. Finalmente, el registro se escribe en el disco, mientras que la clave correspondiente se inserta en el vector "ENTRY" como se indica en la figura 5, donde "K" indica el número de elementos existentes en el vector. Como puede constatarse, se efectúa un desplazamiento de las claves insertadas con anterioridad, hasta encontrar el lugar para la nueva, de modo que se mantenga la ordenación.

Si se solicita otra inserción se repite la operación con alguna modificación, es decir, se controla la disponibilidad de memoria en disco y el vector entry, obligando ocasionalmente a una fusión o intercalación, escribiendo los índices en disco, etc., hasta la conclusión de la fase.

Vale la pena examinar el algoritmo de **fusión** (merge) utilizado (descrito en la figura 6), que encuentra correspondencia en un número verdaderamente reducido de instrucciones BASIC. NX indica el número de elementos del vector índice antes de la inserción, "K" el número de los elementos a insertar y EA el número de elementos asignados. Una vez actualizado el número de elementos asignados se ejecuta la operación de inserción tantas veces como elementos haya que insertar. Al estar ordenados los vectores index y entry es posible, a partir de los últimos elementos del vector index, efectuar desplazamientos y comparaciones con el último elemento del vector entry, transfiriendo este último solamente cuando resulte ser mayor que la clave comparada.

De este modo se prosigue el proceso hasta agotar los elementos del vector entry con una sola exploración del vector in-

1.1
ARCHIVO LLENO 20010
SI
NO
INDICACION DE QUE LOS INDICES AUXILIARES YA NO ESTAN ACTUALIZADOS K=1 CM=0 20020
INICIALIZACION DEL VECTOR ENTRY (2400) 20050
SOLICITUD DE LOS DIVERSOS CAMPOS DEL REGISTRO (1500) 1.1.3 20120
COMPROBACION DE QUE LA CLAVE YA EXISTE CON ANTERIORIDAD (23000) 20130
FUSION DEL VECTOR 20150
ESCRITURA EN DISCO DEL REGISTRO (3100) 1.1.1 20200
VISUALIZACION DE "YA NO HAY LUGAR" (59000)
¿OTROS REGISTROS?
NO
SI
NX+K==EM
SI
NO
VISUALIZA "YA NO HAY LUGAR" (59000)
K=25
NO
SI
ENTRY EN EL VECTOR INDEX (2500) 1.1.2
ACTUALIZ. DE INDEX EN DISCO (3000)
ESCRIT. DEL FICHERO BD. CTRL (2000)
ENTRY EN EL VECTOR INDEX (2500) 1.1.2
ACTUALIZ. DE INDEX EN DISCO (3000)
ESCRIT. DEL FICHERO BD. CTRL (2000)
INICIALIZ. EN EL VECT. ENTRY (2400)
K=0
K=K+1

```
20000 REM INSERCION DEL REGISTRO EN EL ARCHIVO BASE
20010 IF NX=EM THEN  GOSUB 59000:RETURN
20020 FOR  I=1 TO 4:IA(I,2)=0:NEXT
```



```
20050 GOSUB 24000
20060 K=1
20070 CM=0:J=0
20080 HOME:HTAB 15:PRINT "INSERCION"
20090 FOR I=1 TO NC(0):RV$(I,0)="":NEXT
20120 GOSUB 15000
20150 PRINT W$(1);EY%(K)
20160 FOR I= 1 TO NC(0):PRINT RV$(I,0):NEXT
20170 PRINT D$
20200 GOSUB 3100
20210 VTAB 23:CALL -968:PRINT "OTROS REGISTROS?";:GET A$:PRINT :IF A$="N" THEN 2
0260
20230 IF NX+K=EM THEN GOSUB 59000:GOTO 20260
20240 IF K=25 THEN GOSUB 2500:GOSUB 2000:GOSUB 24000:K=0
20250 K=K+1:GOTO 20080
20260 GOSUB 2500:GOSUB 2000
20270 PRINT C$(1):PRINT D$(1)
20280 RETURN
24000 REM INICIALIZACION ENTRADA
24005 EY$(0)=""
24010 FOR I=1 TO 25
24020 EY$(I)=FR$:EY%(I)=NX+I
24030 IF EA >= NX+I THEN PRINT R$(0)(NX+I):INPUT A$:INPUT EY%(I):PRINT D$
24040 RA=DL:DL=RO
24050 RETURN
```

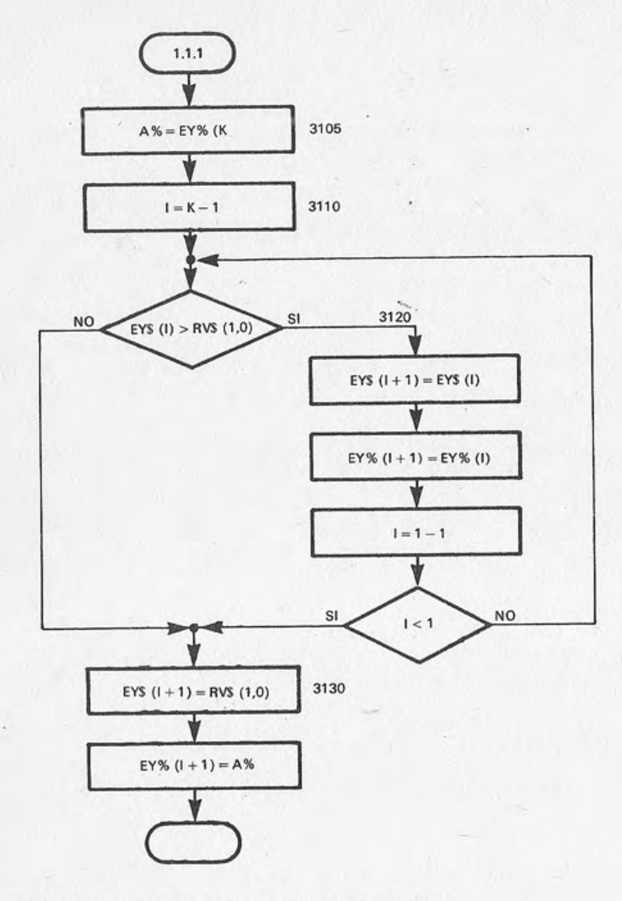

Figura 4.—Inserción del registro en el archivo base.

dex. De este modo se obtiene también el índice a partir del cual hay que iniciar la reescritura en disco de los únicos elementos objeto de actualización.

Borrado

La fase permite el borrado de registros base y listas correspondientes anteriormente objeto de inserción. Es inútil decir cuán peligrosa es dicha función, por lo que necesita la confirmación por parte del operador para permitirle darse cuenta perfecta de lo que va a hacer.

En las búsquedas el acceso a los datos de base se realiza siempre a partir del índice principal o de los índices auxiliares con el puntero asociado a los mismos. Para borrar el registro deseado basta inhibir el acceso al registro base, es decir, eliminar

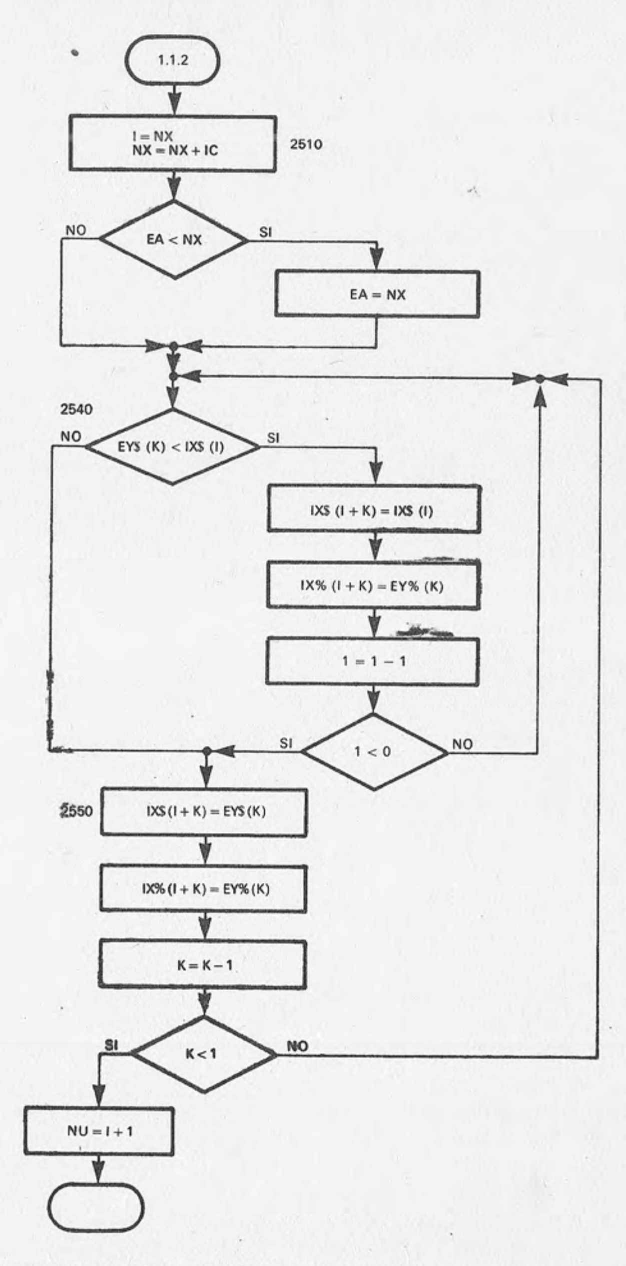



```
3100 REM INSERCION DE UNA NUEVA CLAVE EN EL VECTOR ENTRY
3105 A%=EY%(K)
3110 FOR I=K-1 TO 1 STEP -1
3120 IF EY$(I) > RV$(1,0) THEN EY$(I+1) = EY$(I):EY%(I+1)=EY%(I):NEXT
3130 EY$(I+1)=RV$(1,0):EY%(I+1)=A%
3140 RETURN
```


Figura 5.—Inserción de una nueva clave en el vector ENTRY.

simplemente la clave del índice principal e indicar la no validez de los índices auxiliares o secundarios.

En lo que respecta a la ocasional lista asociada, bastará escribir 0 en el fichero LNK.ENTRY y poner en cola la lista con la lista libre.




```
2500 REM INSERCION DEL VECTOR ENTRY EN EL VECTOR INDEX
2505 VTAB 23:HTAB 1:PRINT "ORDENACION EN CURSO. ATENCION!"
2510 I=NX:NX=NX+K:IF EA < NX THEN EA=NX
```

```
2520 FOR K=K TO 1 STEP -1
2530 FOR I=I TO 1 STEP -1
2535 PRINT R$(0)I:INPUT IX$:INPUT IX%:PRINT W$(0)(I+K)
2540 IF EY$(K) < IX$ THEN PRINT IX$:PRINT IX%:NEXT
2550 PRINT EY$:PRINT EY%(K)
2560 NEXT K
2570 PRINT C$(0):PRINT D$(0)
2580 RETURN
```

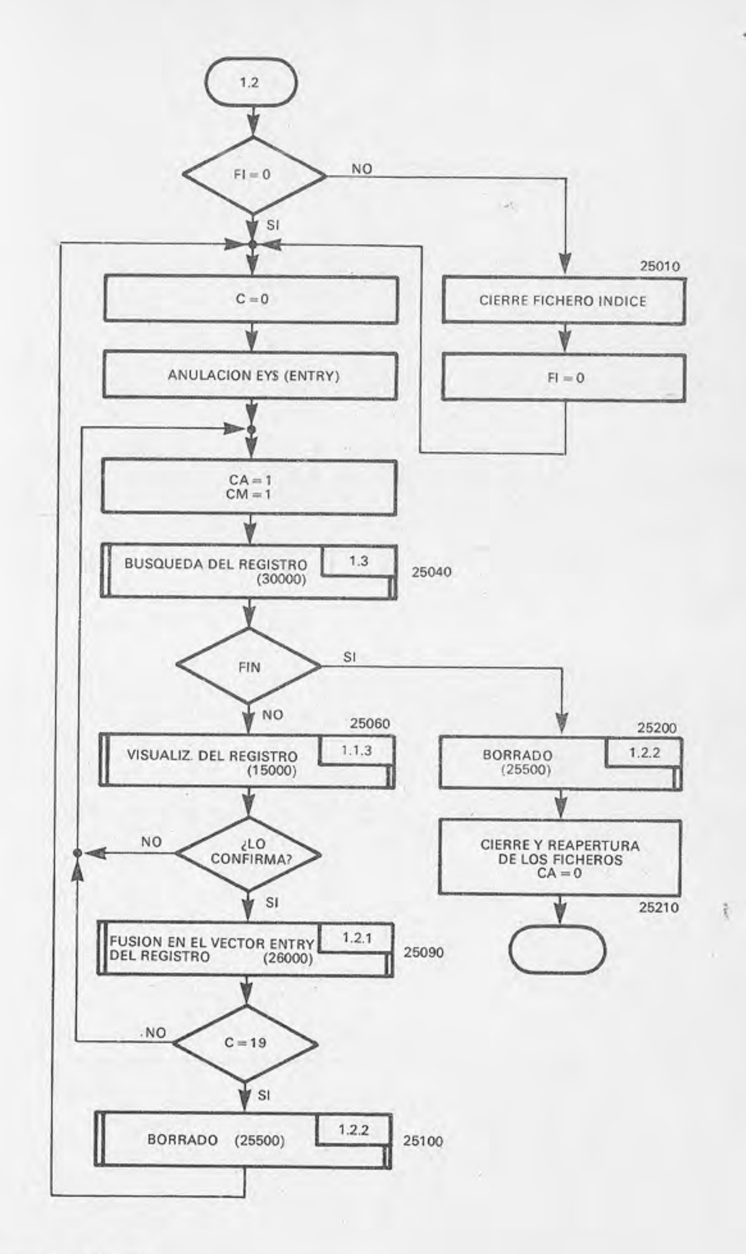


Figura 6.—Fusión del vector ENTRY en el vector INDEX.

Para todos los efectos el registro base no se elimina de manera física, sino que se declara libre la llave de acceso correspondiente, con lo que la primera inserción dispondrá del espacio por superposición.

Las modificaciones que el borrado exige al diccionario principal son opuestas a las que requiere la inserción, pero llevan consigo también la actualización del MST.INX en el disco, con todas las problemáticas de velocidades antes consideradas. Por consiguiente, utilizaremos también en este caso los vectores de preparación EY$ y EY%, donde almacenar claves y punteros a los registros objeto de borrado, y solamente al final de la fase, o al llenarse el vector, exploraremos una vez el vector IX$ para anular, de forma lógica, las claves y ponerlas a la cola de las claves válidas reordenadas.

Haciendo referencia a la figura 7, si se habilitó un índice secundario se obliga al empleo del índice principal, porque las actualizaciones sucesivas se efectuarán solamente en dicho índice. A continuación, se preparan los vectores **entry** y se pone a 0 el número de registros objeto de borrado. Para solicitar al operador qué registro borrar se utiliza parte de la rutina de búsqueda, a la que se pasa el parámetro CA = 1. Una vez encontrado el registro objeto de borrado, se visualizará y se solicitará la confirmación correspondiente con miras a la seguridad. En caso afirmativo, se almacena la clave en el vector entry de manera ordenada aumentando el número de registros objeto de borrado. Solamente cuando sean 19, o cuando el operador ya no solicite más eliminaciones, se efectuará la compactación en el vector IX$ de las claves principales válidas y la puesta en cola de las claves anuladas. Para cerciorarse de que cada información se almacenó en disco, después de la escritura se efectúa un cierre y una posterior reapertura de los ficheros utilizados. Los bloques correspondientes a la búsqueda de registro (1.3) y visualización (1.1.3) los analizaremos




```
25000 REM ELIMINACION
25010 IF FI THEN PRINT C$(XF):FI=0
25020 C=0
25030 FOR I=0 TO 20:EY$(I)=FP$:EY%(I)=0:NEXT
```

```
25040 CA=1:GOSUB 30000
25050 IF KM$="" THEN 25200
25060 CM=1:J=0:HOME:GOSUB 15000
25070 VTAB 22:INPUT "LO CONFIRMA? ";B$
25080 IF B$<> "S" THEN 25040
25090 GOSUB 26000
25100 IF C=19 THEN GOSUB 25500:GOTO 25020
25130 GOTO 25040
25200 GOSUB 25500
25210 PRINT C$(0)
25230 PRINT O$(0)
25240 IF FL THEN FOR I=4 TO 6:PRINT C$(I):PRINT O$(I):NEXT
25280 EY$(0)="":CA=0
25290 RETURN
```

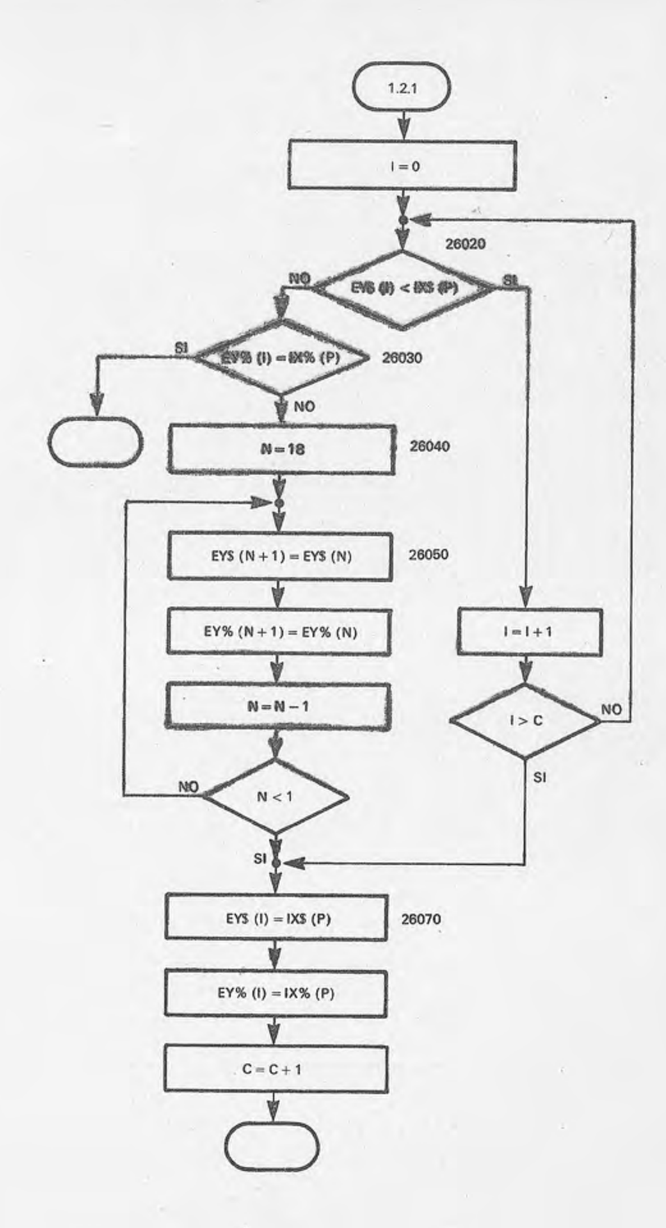
Figura 7.—Borrado de registro en archivo base.

a continuación, mientras que veremos con detalle la fusión de la clave en el vector entry y el borrado.

La figura 8 representa la inserción de la clave en el vector entry. En primer lugar, se explora el vector entry hasta encontrar un elemento mayor o igual a la clave IX$ a insertar; si la clave y el elemento son iguales, el registro está ya en la fase de borrado y se llega, pues, a una situación no variada. Por el contrario, si el elemento es mayor, se trasladan todos los elementos sucesivos una posición, dejando espacio para la inserción de la clave, aumentando el número de elementos objeto de borrado.

En la figura 9 mostramos la **comprensión** del diccionario frente a las claves objeto de borrado. Es evidente, que si no hay borrados nada se modificará; de no ser así, se visualizará una lista de los registros que han de borrarse, solicitando una posterior confirmación por parte del operador. En caso afirmativo se borrarán antes las cadenas asociadas a los registros y luego se actualizará el índice principal según se indica en la figura 10.

Para hacer más rápido el procedimiento se encuentra la primera clave con la búsqueda binaria y se procede luego a reubicaciones en compresión, recubriendo las claves a eliminar, hasta silvar todos los elementos válidos. Para poder utilizar los registros borrados se asocian a las claves libres FR$ los punteros a los registros, almacenados de forma provisional en el vector EY%. Por último, se actualiza el número de los registros válidos y se indica que están fuera de uso los índices auxiliares o secundarios.




```
26000 REM INSERCION DEL REGISTRO ELIMINADO EN EL VECTOR ENTRY
26010 FOR I=0 TO C
26020 IF EY$(I) < IX$ THEN NEXT
26030 IF EY%(I) = IX% THEN RETURN
26040 FOR N=18 TO I STEP -1
```

```
26050 EY$(N+1)=EY$(N):EY%(N+1)=EY%(N)
26060 NEXT
26070 EY$(I)=IX%:EY%(I)=IX%
26080 C=C+1:RETURN
```


Figura 8.—Fusión en el vector ENTRY del registro borrado.

Búsqueda binaria

Con respecto al algoritmo propiamente dicho de búsqueda binaria presentando en el capítulo segundo, la subrutina de la figura 11 resulta más complicada, por cuanto que controla la búsqueda tanto en memoria (en el vector de los índices principales) como en el disco (en los ficheros de los índices secundarios).

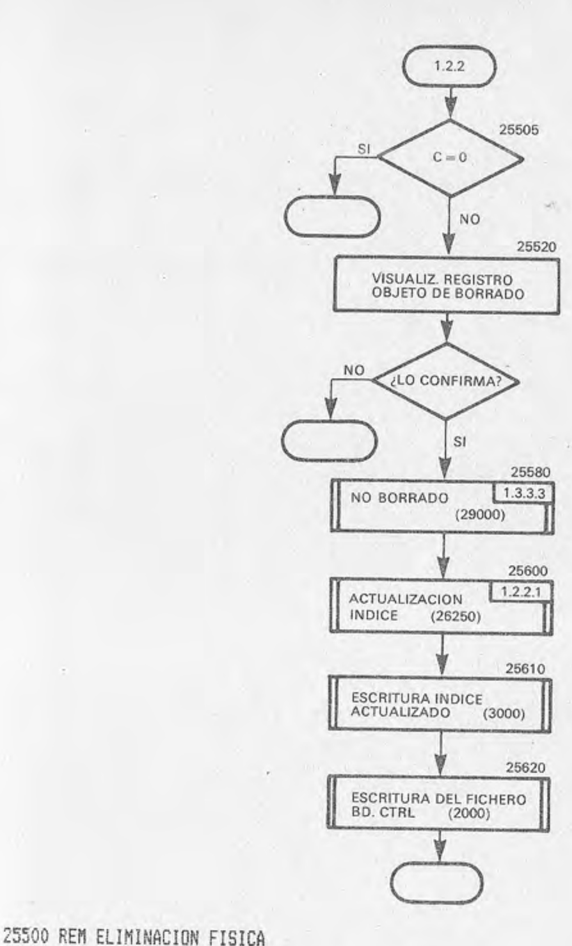
Además, la comparación se efectúa solamente en los LK primeros caracteres del campo, en donde LK es la longitud de la clave, con el fin de encontrar todas las claves cuyos primeros caracteres sean iguales a la clave buscada.

Después de haber asignado los valores iniciales a las variables, si se están utilizando los índices auxiliares se toma IV como el número máximo de elementos y el menor entre la longitud de la clave objeto de búsqueda y la longitud de la clave grabada en disco (puede suceder si la clave en disco fue objeto de truncación). A continuación, la subrutina busca la clave en modo binario, comparando KM$ (clave solicitada) con A$, que contiene una de las claves en memoria o en disco, según el tipo de índice utilizado. Si a la salida "F" vale 0, ello indica que la clave no fue encontrada y si vale 1, en "P" existe el número de orden de la clave en búsqueda en el vector adecuado.

Cambio de índice

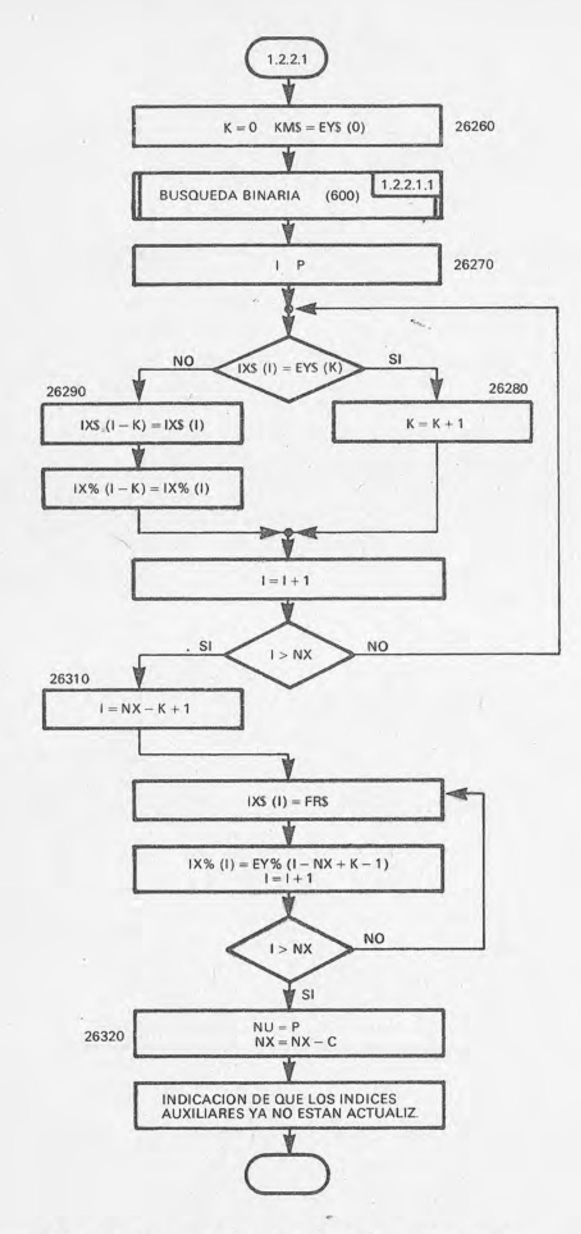
La figura 12 representa la subrutina que permite el índice de acceso al fichero MST.FL. Estos índices auxiliares están grabados en disco, por lo que la búsqueda binaria ya no se efectúa en memoria, sino en disco. La variable FI indica, si es diferente de 0, cuáles de los cuatro índices posibles se están utilizando.

A la entrada de la subrutina, si era activo cualquier índice secundario, se cerrará el fichero correspondiente y se presentará la situación de los índices disponibles, con la indicación de si están




```
25500 REM ELIMINACION FISICA
25505 IF NOT C THEN RETURN
25510 HOME:HTAB 10:PRINT "REGISTRO EN ELIMINACION":PRINT
25520 FOR I=0 TO C-1:PRINT EY$(I):NEXT
25530 PRINT :INPUT "LO CONFIRMA? ";A$
25540 IF A$<>"S" THEN  RETURN
25560 FOR I=0 TO C-1
25570 N=EY%(I)
25580 IF FL THEN GOSUB 29000
25590 NEXT
25600 GOSUB 26250
25620 GOSUB 2000
25630 RETURN
```


Figura 9.—Eliminación física del registro.




```
26250 REM COMPACTACION INDICES ANTIGUOS (ACTUALIZACION DEL INDICE)
26255 PRINT :PRINT "COMPACTACION: ";:FLASH:PRINT "ATENCION":NORMAL
26260 K=0:KM$=EY(0):GOSUB 600:GOSUB 3500
26270 PRINT R$(0)I:INPUT IX$:INPUT IX%
```



```
26280 IF IX%=EY%(K) THEN K=K+1:GOTO 26300
26290 PRINT W$(0)(I-K):PRINT IX$:PRINT IX%
26300 NEXT
26310 FOR I=EA-K+1 TO EA:PRINT W$(0)I:PRINT FR$:PRINT EY%(I-EA+K-1):NEXT
26320 NX=NX-C:FOR I=1 TO 4:IA(I,2)=0:NEXT
26330 RETURN
```


Figura 10.—Actualización del índice después del borrado.

actualizados o no, solicitando cuál se desea utilizar. Una vez realizada la elección se abre el fichero correspondiente y a partir del mismo se lee el número de elementos existentes en el fichero, que puede ser diferente de NX si el índice ya no está actualizado. Este número (IV) se utilizará en las subrutinas de búsqueda en lugar de NX como límite superior de búsqueda.

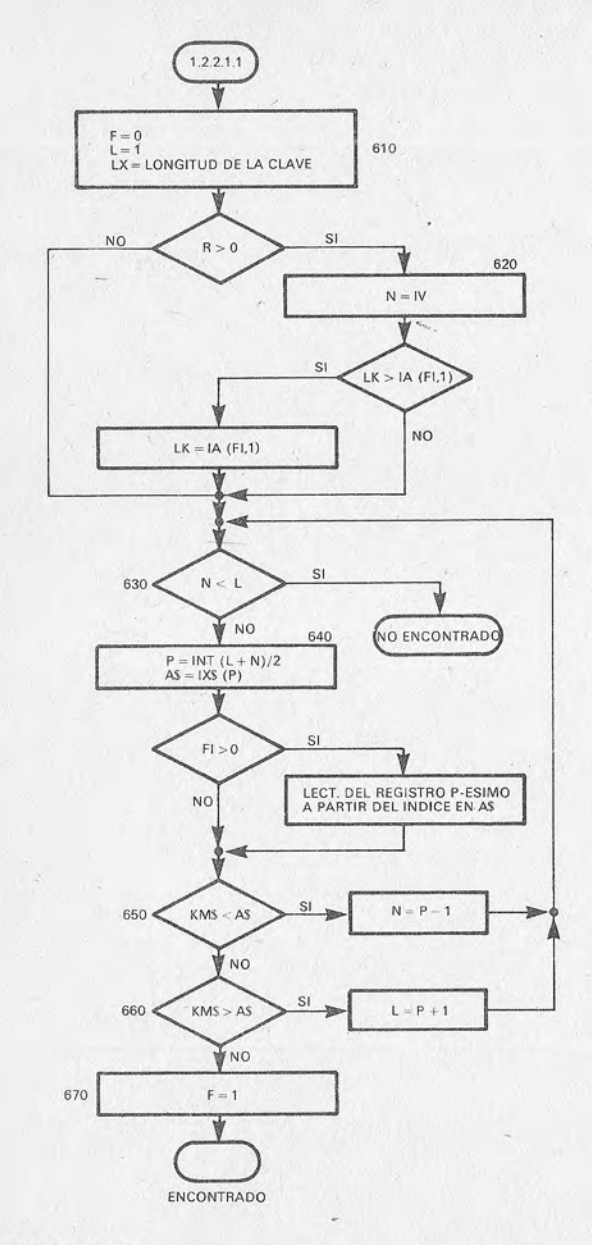
Habilitación-deshabilitación de cadena

La subrutina de la figura 13 efectúa la negación del estado anterior de utilización, correspondiente a las cadenas, condicionado no obstante por el indicador PL de presencia de Link. De hecho, si existen los ficheros concatenados (Link) y están activos (PL y FL puestos a 1), se cerrarán los ficheros correspondientes. Si PL está puesto a 1 a FL se le asigna el valor negado del anterior. Si FL pasa a 1, ello quiere decir que las cadenas estaban desactivadas y se quiere habilitarlas; entonces se lee el fichero LNK.CTRL, que contiene la descripción de los registros concatenados, y se abren los tres ficheros necesarios para el acceso a los datos.

Búsqueda y/o modificación

La subrutina de búsqueda de un registro es el elemento fundamental del programa, por cuanto que permite tener acceso, de diversos modos, a los registros de archivo, así como por permitir la gestión completa de los ficheros concatenados; además, se utiliza también en parte de la fase de borrado.

Inicialmente, según puede observarse en la figura 14, se solicita la **clave,** que es el primer campo del registro si se está utilizando el fichero MASTER INDEX o si se está empleando el cam-

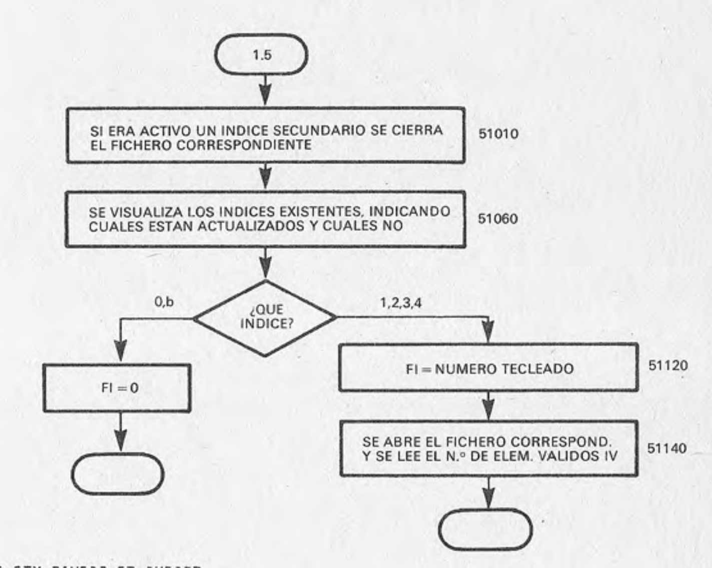



```
600 REM BUSQUEDA BINARIA
610 F = 0: L = 1: N = NX: LK = LEN (KM$)
620 IF FI THEN N = IV:IF LK > IA(FI,1) THEN LK = IA(FI,1)
630 IF N < L THEN RETURN : REM NO ENCONTRADO
```



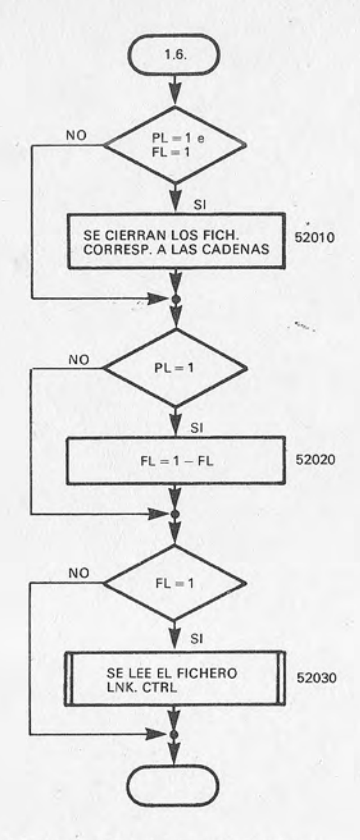
```
640 P = INT ((L + N) / 2):PRINT R$(XF);P:INPUT A$:PRINT D$
650 IF LEFT$ (KM$,LK) < LEFT$ (A$,LK) THEN N = P - 1:GOTO 630
660 IF LEFT$ (KM$,LK) > LEFT$ (A$,LK) THEN L = P + 1:GOTO 630
670 F = 1:RETURN: REM ENCONTRADO
```


Figura 11.—Búsqueda binaria.




```
51000 REM CAMBIO DE INDICE
51010 IF FI THEN PRINT C$(FI+6)
51030 HOME:HTAB 15:PRINT "INDICES POSIBLES":PRINT :PRINT "0 ";LB$(1,0)
51040 FOR I=1 TO 4
51050 IF NOT IA(I,0) THEN  51080
51060 PRINT I" ";LB$(IA(I,0),0);:IF NOT IA(I,2) THEN HTAB 20:PRINT "NO";
51070 HTAB 24:PRINT "ACTUALIZADO"
51080 NEXT
51090 VTAB 10:INPUT "CUAL? ";A$:A%=VAL(A$)
51100 IF NOT A% THEN FI=0:XF=FI:RETURN
51110 IF A% >4 OR IA(A%,0)=0 THEN 51090
51120 FI=A%:XF=FI+6
51130 PRINT O$(XF)
51140 PRINT R$;0:INPUT IV:PRINT D$
51170 RETURN
```

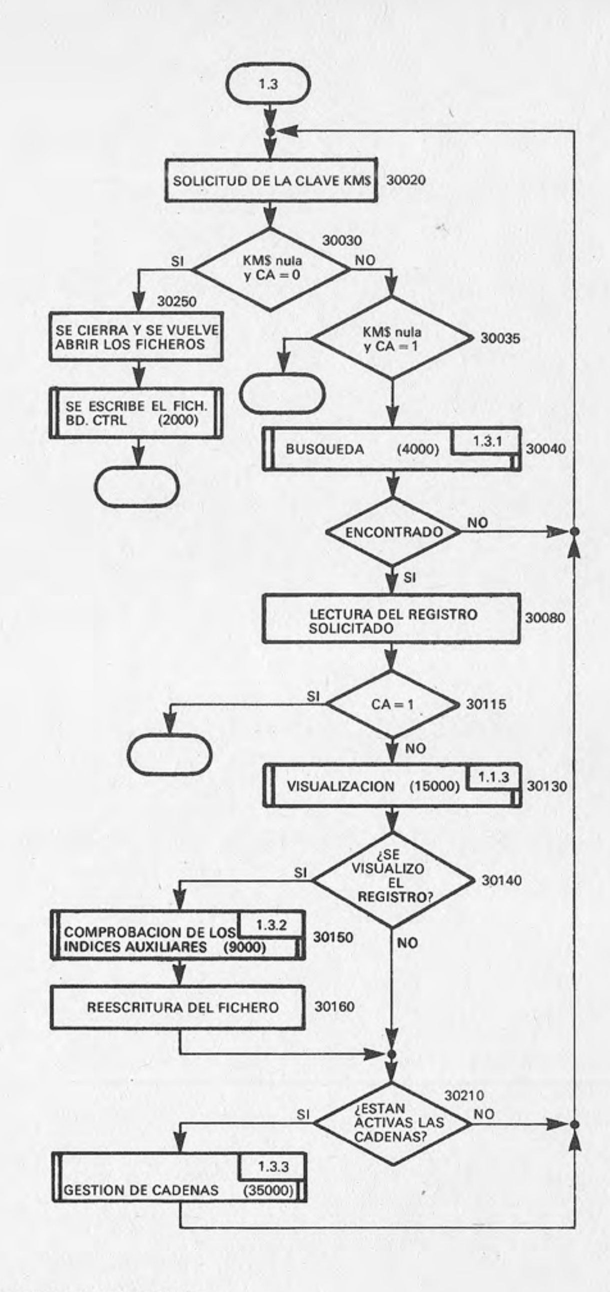

Figura 12.—Cambio del índice.




```
52000 REM GESTION DE INDICADORES DE ESTADO DE LAS CADENAS
52010 IF PL AND FL THEN FOR I=4 TO 6:PRINT C$(I):NEXT
52020 IF PL THEN FL=NOT FL
52030 IF FL THEN GOSUB 10000
52040 RETURN
```


Figura 13.—Habilitación/deshabilitación de las cadenas.

po correspondiente al índice auxiliar o secundario. Para el caso de clave nula se sale de la subrutina, cerrando los ficheros si se efectuó alguna modificación (para asegurarlos contra la caída de tensión) y escribiendo en el fichero BD.CTRL la nueva situación (lo que es necesario para indicar que un índice auxiliar ya no está actualizado si se modificó el campo correspondiente en uno cualquiera de los registros).




```
30000 REM BUSQUEDA Y MODIFICACION
30010 HOME
30020 PRINT LB$(1+(FI)0)*(IA(FI,0);:INPUT ": ";KM$
```

```
30030 IF KM$="" AND (NOT CA) THEN 30250
30035 IF KM$="" AND (CA) THEN RETURN
30040 GOSUB 4000:IF NOT F THEN  30000
30060 GOSUB 700
30100 FOR I=1 TO NC(J):RB$(I,J):NEXT
30110 IF (CA) THEN  RETURN
30120 CM=2-FL:HOME:GOSUB 15000
30140 IF NOT FM THEN  30210
30150 GOSUB 9000
30160 PRINT W$(1);IX%
30170 FOR I=1 TO NC(0):PRINT RV$(I,0):NEXT :PRINT D$
30210 IF FL THEN N=IX%:GOSUB 35000
30220 GOTO 30000
30250 IF SL(0) THEN PRINT C$(1):PRINT O$(1):GOSUB 2000
30270 IF SL(1) THEN FOR I=4 TO 6:PRINT C$(I):PRINT O$(I):NEXT
30290 RETURN
```


Figura 14.—Búsqueda y modificación.

Por el contrario, si la clave no es nula se inicia la subrutina de búsqueda propiamente dicha (a partir de la línea 4000) y si se encuentra el registro será objeto de lectura. Después de lo anterior si la etapa fue actuada por el borrador (CA=1), se saldrá de la subrutina. De no ser así se visualizará el registro, permitiendo la modificación de cada campo con la excepción de la clave principal. Si se efectúan modificaciones se inicia una subrutina (líneas a partir de la 9000) que sirve para comprobar si se varió un campo utilizado como índice (en cuyo caso, se señala que el índice ya no está actualizado) y luego se vuelve a escribir el registro. En éste punto, si están activos los registros de Link, se llama la subrutina de gestión correspondiente (líneas a partir de la 35000).

CAPITULO VII

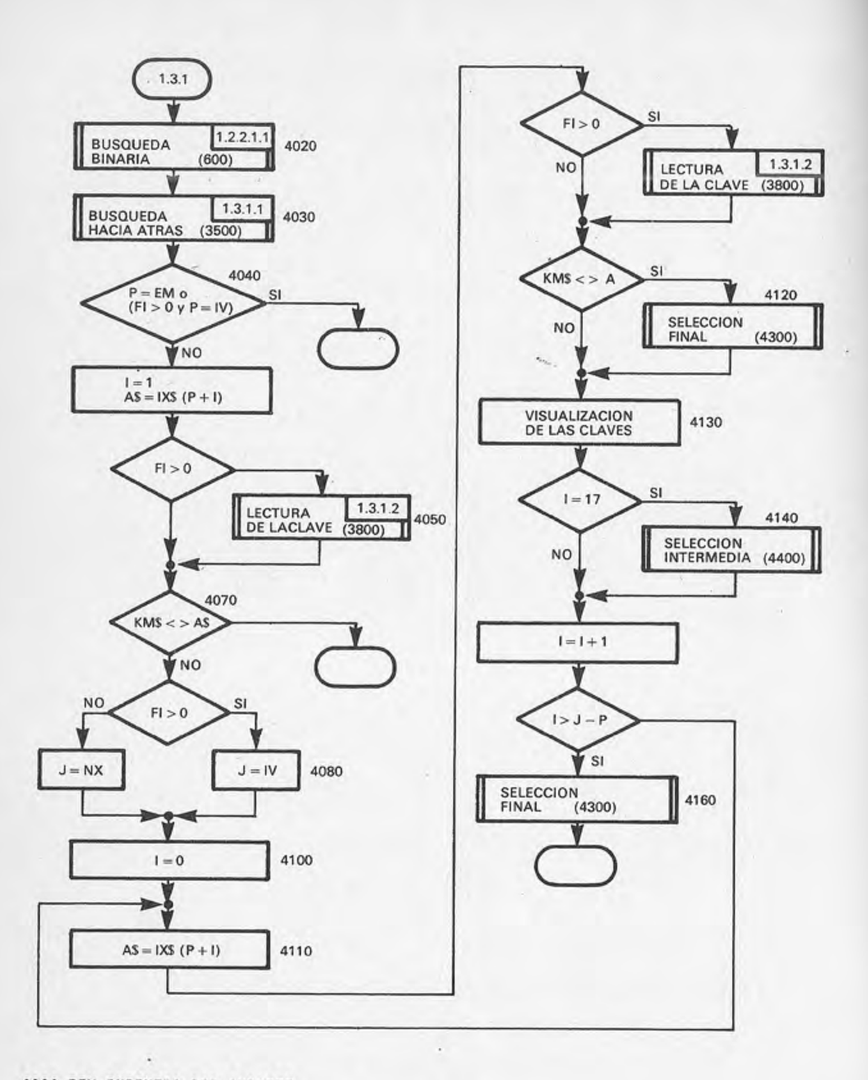
ULTIMAS SUBRUTINAS...


Podemos observar en la figura 1 cómo se efectúa la búsqueda: la subrutina de búsqueda binaria, entre todas las claves cuyos primeros caracteres sean iguales a la de la clave buscada, proporciona el índice de una de ellas, pero no se sabe si es la primera, la última o una cualquiera de ellas. Es necesario, pues, partiendo de la clave señalada por la búsqueda binaria remontarse hacia atrás en el diccionario (figura 2) hasta encontrar una clave que no sea igual a la de búsqueda. Después de ello, si el elemento encontrado es el último existente en el índice en uso, terminará la subrutina.

En resumen, pues, la búsqueda se realiza del modo siguiente: proporcionada la clave de acceso, la subrutina de búsqueda binaria suministra el índice de una clave cuyos primeros caracteres son iguales a los de la clave requerida; luego se busca la primera de estas claves y a partir de ella en adelante se visualizan todas las que satisfacen la condición requerida, para pasar al operador la deseada. Se ve pues que, cualquiera que sea el índice que se esté utilizando, basta proporcionar al programa una clave parcial para que se encuentren todas aquéllas cuyos primeros caracteres sean idénticos a la clave requerida.

En la figura 4 constatamos lo que hace la subrutina de control con respecto a los índices auxiliares o secundarios: en efecto, para cada uno de los índices existentes comprueba si se modificó el campo que hace de índice, en cuyo caso señala que el índice ya no está actualizado (IA(I,2)=0).

De no ser así se visualizan todas las claves compatibles con la buscada, subdivididas en grupos para permitir la elección de




```
4000 REM BUSQUEDA POR INDICES
4010 B$="SELECCIONAR POR NUMERO":E$="RETURN PARA "
4020 GOSUB 600:GOSUB 900
4030 GOSUB 3500:GOSUB 900
4040 IF (P=EM) OR (FI AND P=IV) THEN RETURN
4050 I=1:GOSUB 3800
4070 IF KM$<> LEFT$ (A$,LEN(KM$)) THEN RETURN
4080 J=NX:IF FI THEN J=IV
4090 PRINT
4100 FOR I=0 TO J-P
4110 GOSUB 3800
```



```
4120 IF KM$<>LEFT$(A$,LEN(KM$)) THEN GOSUB 4300
4130 PRINT SPC(3-LEN(STR$(I+1)));I+1" "A$;
4134 IF FI THEN  HTAB (39-LL(1,0):PRINT " "KO$;
4136 PRINT
4140 IF I=INT (I/17)*17 AND I THEN GOSUB 4400
4150 NEXT
4160 GOSUB 4300:RETURN
4300 REM SELECCION FINAL
4310 VTAB 22:PRINT B$
4320 PRINT E$;:INPUT "SALIR";A$:A%=VAL(A$)
4330 IF NOT A% THEN F=0:GOTO 4360
4340 IF A%<1 OR A%>I THEN 4310
4350 P=P+A%-1
4360 POP:RETURN
4400 REM SELECCION MEDIA
4410 VTAB 22:PRINT B$
4420 PRINT E$;:INPUT "CONTINUAR";A$:A%=VAL(A$)
4430 IF NOT A% THEN VTAB 3:CALL -958:RETURN
4440 IF A%<1 OR A%>I+1 THEN 4410
4450 P=P+A%-1
4460 POP:RETURN
```


Figura 1.—Búsqueda mediante los diversos índices.

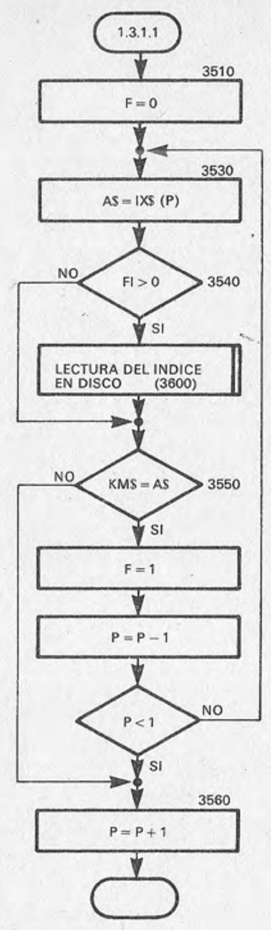
la deseada. Dos son las subrutinas utilizadas con tal objeto; una es la de selección intermedia, cuando la pantalla está llena aunque existen todavía claves compatibles, y otra es la de selección final cuando, por el contrario, se han visualizado todas las claves. Si se está utilizando un índice auxiliar o secundario se leerá la clave principal a partir del registro base (figura 3) y si la clave del índice estaba abreviada, también la clave completa, visualizando luego ambos campos.

Pasamos ahora a la parte más árida, a la más compleja del programa: la gestión de los registros concatenados.

En la figura 5 se representan las acciones que se pueden tomar con respecto a las listas: inserción, exploración y borrado de registros, con eliminación completa de una cadena.

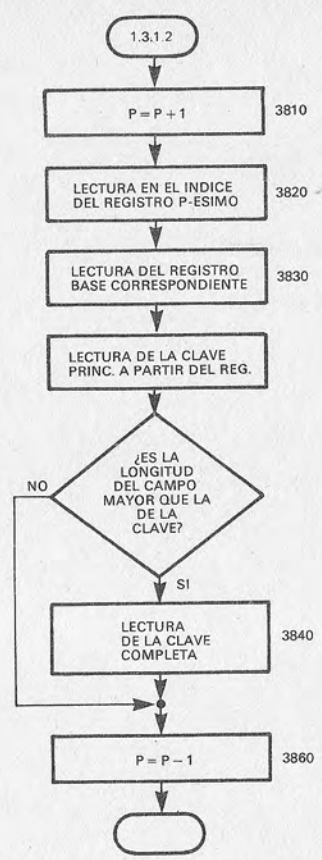
Pero no acaba aquí la cuestión, puesto que es preciso completar todavía la actualización de la lista correspondiente al registro base en el que se está trabajando.

Se lee luego a partir de LINK.ENTRY y la antigua cabecera de la lista (que será 0 si no había registros concatenados con anterioridad) y se escribe en el fichero LNK.PTR, en correspondencia




```
3500 REM BUSQUEDA HACIA ATRAS
3510 F=D
3520 FOR P=P TO 1 STEP -1
3540 GOSUB 3600
3550 IF KM$=LEFT$(A$, LEN,(KM$)) THEN  F=1:NEXT
3560 P=P+1
3570 RETURN
3600 REM LECTURA DE INDICE EN DISCO
3610 PRINT R$(XF);P:INPUT A$:INPUT A%
3620 IF FI THEN IF LEN(KM$) > IA(FI,1) THEN PRINT R$(1);A%:FOR I=1 TO IA(FI,0):I
NPUT A$:NEXT
3630 PRINT O$
3640 RETURN
```

 *Figura 2.—Búsqueda hacia atrás.*

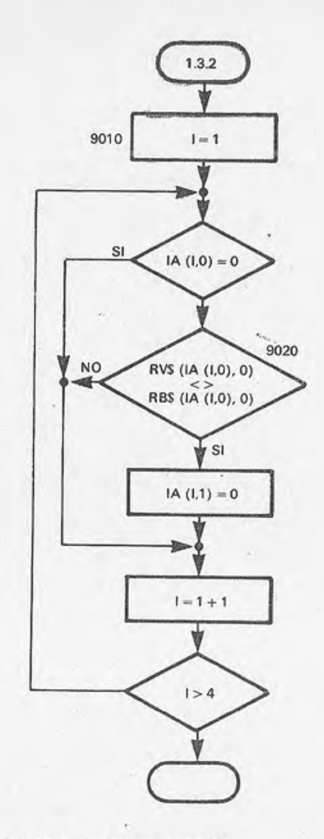



```
3800 REM LECTURA EN DISCO DE LA CLAVE DE ACCESO A PANTALLA
3810 P=P+1
3820 PRINT R$(XF);P:INPUT A$:INPUT A%:IF NOT FI THEN  3850
3830 PRINT R$(1);A$:INPUT KO$
3840 IF IA(FI,1) < LL(IA(FI,0)0) THEN FOR HK=2 TO IA(FI,0):INPUT A$:NEXT
3850 PRINT D$
3860 P=P-1
3870 RETURN
```

 *Figura 3.—Lectura de la clave de acceso en disco.*

con el registro escrito en el fichero LNK.FILE; por último se escribe en el fichero LNK.ENTRY el número del registro LNK.FILE.

En todo el procedimiento "N" indica el número del registro base en el que se está trabajando y "X" el número del registro de cadena que se tiene que escribir.

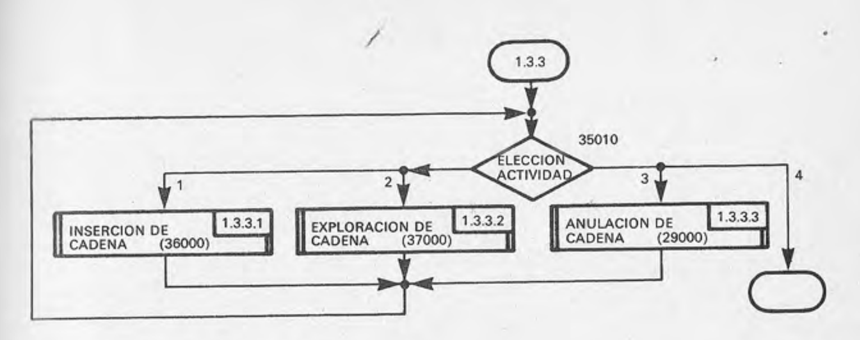



```
9000 REM BUSQUEDA VARIACIONES DE LOS CAMPOS INDICES
9010 FOR I=1 TO 4
9020 IF IA(I,0) THEN IF RV$(IA(I,0),0) <> RB$(IA(I,0),0) THEN IA(I,2)=0
9030 NEXT
9040 RETURN
```

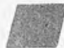

Figura 4.—Búsqueda de variaciones de los campos índice.

Inserción

Se comprueba (figura 6) que el indicador DF que señala la condición de disco lleno no es igual a 1; de no ser así se termina de inmediato la fase. En caso contrario se prosigue la ejecución del programa llamando a la subrutina de entrada de campos ge-



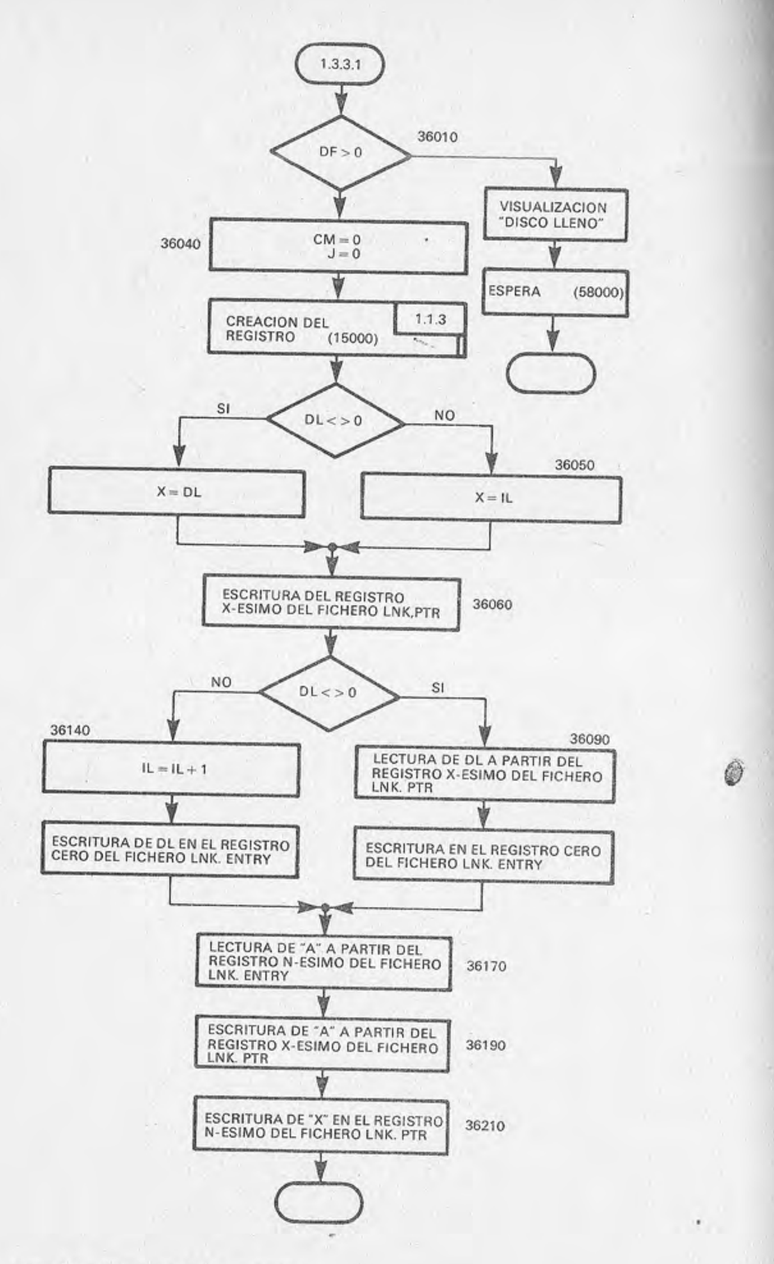

```
35000 REM GESTION DE LAS CADENAS
35010 VTAB 19:CALL -958:PRINT " 1- INSERCION EN CADENA"
35020 PRINT " 2- EXPLORACION Y ELIMINACION"
35025 PRINT " 3- ELIMINACIN CADENA"
35030 INPUT "(RETURN PARA SALIR) ";A$
35040 IF A$="" THEN RETURN
35050 ON VAL(A$) GOSUB 36000,37000,29000
35060 GOTO 35000
```


***Figura** 5.—Gestión de las cadenas.*

neralizada (líneas a partir de 15000) con parámetros CM=0 que indica la creación y J=1 que indica que se trata del registro de cadena. Una vez terminada la entrada de los campos es preciso escribir en el disco, y si la cabecera de la lista libre DL es nula, se escribirá el registro número IL; en caso contrario se escribirá el registro número DL. En el primer caso se tiene que incrementar IL y escribirlo en el registro 0 del fichero LNK.PTR; de no ser así, lee a partir del fichero LNK.PTR en el registro que tiene el mismo número que el que se acaba de escribir, la nueva cabecera de la lista libre y la escribe en el registro 0 del fichero LNK.ENTRY.

Exploración y borrado

La subrutina de la figura 7 permite explorar una cadena, visualizando un registro cada vez, modificándolo y, si así se quiere, eliminándolo. La subrutina utiliza el indicador RO (record old = registro antiguo) y el RA (record attale = registro actual) que indican, respectivamente, el número del penúltimo y último registros leídos a partir del disco. Existe una sola limitación, y es que no se pueden borrar dos registros consecutivos. Para hacerlo, después




```
36000 REM INSERCION DEL REGISTRO EN CADENA
36010 IF DF THEN PRINT "DISCO LLENO";:GOSUB 58000:RETURN
36020 ONERR GOTO 38100
```



```
36030 VTAB 5:CALL -958
36040 J=1:CM=0:FOR I=1 TO NC(1):RV$(I,1)="":NEXT :GOSUB 15000
36050 X=DL:IF NOT X THEN X=IL
36060 PRINT W$(6);X
36070 FOR I=1 TO NC(1):PRINT RV$(I,1):NEXT
36080 IF NOT DL THEN 36140
36090 PRINT R$(5);X:INPUT DL
36110 PRINT W$(4);0:PRINT DL
36130 GOTO 36170
36140 IL=IL+1
36150 PRINT W$(5);0:PRINT IL
36170 PRINT R$(4);N:INPUT A
36185 ONERR GOTO 38000
36190 PRINT W$(5);X:PRINT A
36210 PRINT W$(4);N:PRINT X
36230 PRINT D$:POKE 216,0
36240 RETURN
```

Figura 6.—Inserción del registro en cadena.

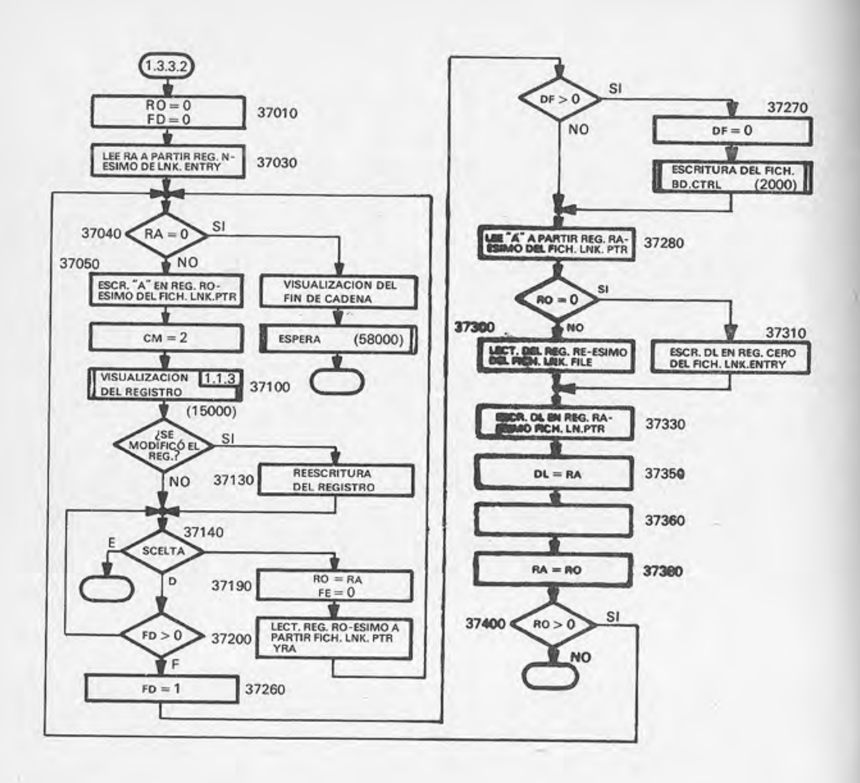


de la primera eliminación, basta volver a comenzar la exploración desde el principio y, una vez alcanzado el nuevo registro a borrar, se puede realizar dicha operación. Por el contrario, si los dos registros no son consecutivos, se pueden borrar libremente sin tener que volver a comenzar desde el principio de la lista.

Cuando se elimina un registro, en la pantalla aparecerá el anterior a no ser que se esté al comienzo de la cadena, en cuyo caso no se visualiza nada.

Durante la exploración de la cadena, en la parte alta de la pantalla se mantienen visualizados los dos primeros campos del registro base, con el fin de saber siempre cuál es la referencia de los datos.

Veámos cómo funciona esta subrutina. Una vez puestos a 0 los indicadores RO y FD (indicador de borrado), se lee en RA la cabecera de la cadena; si RA es igual a cero ello quiere decir que la cadena es nula. Si RA es distinto de cero se lee el registro número RA del fichero LNK.FILE y, con el parámetro CM=2 (modificación) se llama a la habitual subrutina de visualización de registro. A la salida de esta última, si el registro fue objeto de modificación, se le vuelve a escribir y luego, en cada caso, se llega a una posterior posibilidad de elección: salida de la exploración (S), borrado del registro visualizado (B) o continuación de la exploración (RETURN). Si la elección es S terminará la subrutina; si la elección es continuar la exploración, se graba en RO el número de RA




```
37000 REM EXPLORACION DE LA CADENA DE REGISTROS CON POSIBILIDAD DE ELIMINACION
37010 RD=0:FD=0
37020 PRINT R$(4);N
37030 INPUT RA:PRINT D$
37040 IF(NOT RA) THEN VTAB 22:CALL -958:PRINT "FIN ":GOTO 58000
37050 PRINT R$(6);RA
37060 J=1:GOSUB 800
37090 CM=2:VTAB 5:CALL -958
37100 GOSUB 15000
37110 IF NOT FM THEN 37140
37120 PRINT W$(6);RA
37130 FOR I=1 TO NC(1):PRINT RV$(I,1):NEXT
37140 PRINT D$:VTAB 20:CALL -958:PRINT "E)XIT    D)EL."
37150 PRINT "(RETURN PARA CONTINUAR)";:GET A$:PRINT
37160 IF A$="E" THEN RETURN
37170 IF A$="D" AND NOT FD THEN 37250
37180 IF A$<> CR$ THEN 37140
37190 RD=RA:FD=0
37200 PRINT R$(5);RO
```



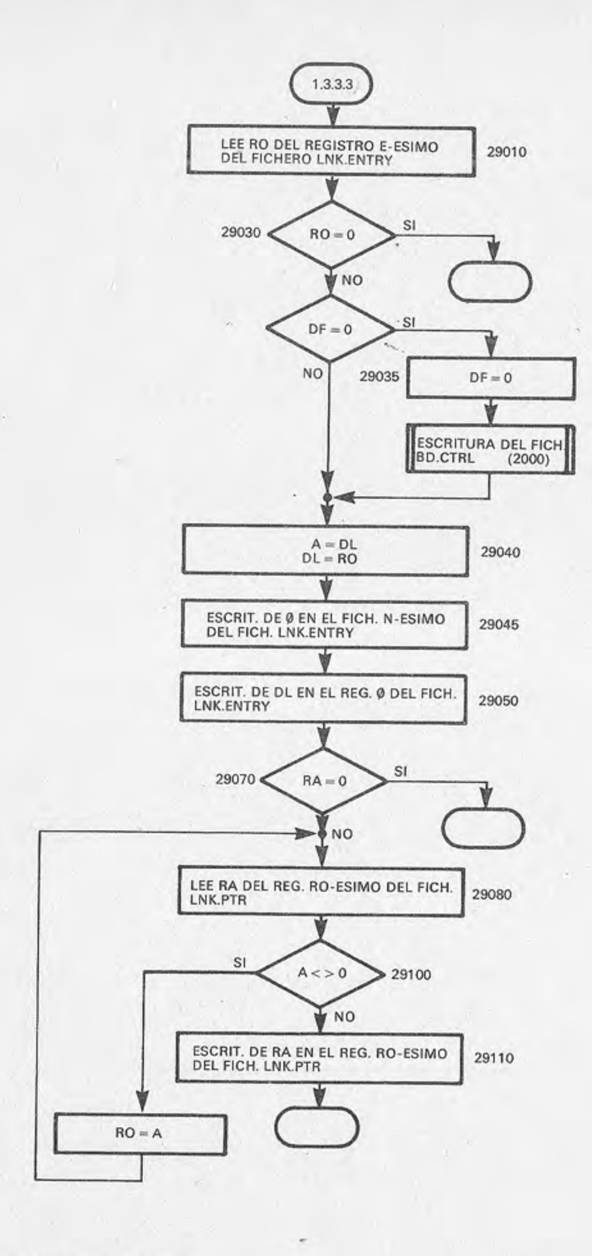
```
37210 GOTO 37030
37250 REM ELIMINACION CADENA
37260 FD=1:SL(1)=1
37270 IF DF THEN DF=0:GOSUB 2000
37280 PRINT R$(5);RA:INPUT A
37290 IF NOT RD THEN  37310
37300 PRINT W$(5);RO:GOTO 37320
37310 PRINT W$(4);N
37320 PRINT A
37330 PRINT W$(5);RA
37340 PRINT DL
37350 DL=RA
37360 PRINT W$(4);0
37370 PRINT DL
37380 VTAB 3:CALL -958:RA=RO
37390 PRINT D$
37400 IF RO THEN 37040
37410 RETURN
```

Figura 7.—*Exploración de la cadena de registros con posibilidad de borrado.*

del registro actual y se lee a partir del fichero LNK.PTR el número RA del siguiente registro que constituye la cadena, volviendo luego al comienzo para comprobar si RA es igual a cero.

Por el contrario, si se solicita el borrado (B) se pondrá FD a 1 para indicar que se efectuó una eliminación y si el disco de los ficheros concatenados estaba lleno, se escribirá en el fichero BD.CTRL que ahora ya no lo está. El borrado se efectúa del modo siguiente: se lee cuál es el número del siguiente registro en la lista y luego, si RO es igual a cero, es decir, si se está al comienzo de la cadena, se escribe este número en el fichero LNK.ENTRY, registro N-ésimo. De no ser así, se escribe en el registro RO del fichero LNK.PTR. Por consiguiente, se consigue que si se elimina el primer registro de la lista se escriba como nueva cabecera el segundo registro de la lista; si se elimina uno cualquiera de los registros se escribirá en el puntero del anterior el número de orden del sucesivo al eliminado.

Ahora sólo nos queda hacer que el registro eliminado pueda reutilizarse, añadiéndolo a la lista libre; entonces se toma la antigua cabecera de esta lista (DL) y se la escribe en el registro eliminado en el fichero LNK.PTR. A DL se le da el valor RA del registro borrado y se escribe en el registro cero de LNK.ENTRY, por




```
29000 REM ELIMINACION EN CADENA
29010 PRINT R$(4);N:INPUT RO
29030 IF NOT RO THEN PRINT D$:RETURN
29035 SL(1)=1:IF DF THEN DF=0:GOSUB 2000
```



```
29045 PRINT W$(4);N:PRINT 0
29050 PRINT W$(4);0:PRINT DL
29070 IF (NOT RA) THEN PRINT D$: RETURN
29080 PRINT R$(5);RO:INPUT A
29100 IF A<>0 THEN RO=A:GOTO 29080
29110 PRINT W$(5);RO:PRINT RA:PRINT D$
29120 RETURN
```


Figura 8.—Eliminación de una cadena.

lo que quedará grabada en disco la nueva situación. Para volver al ciclo normal de la subrutina bastará dar a RA el valor de RO del registro anterior en la lista y volver a la visualización normal si existe el registro o se terminará la subrutina si no hay nada anterior en la lista a visualizar.

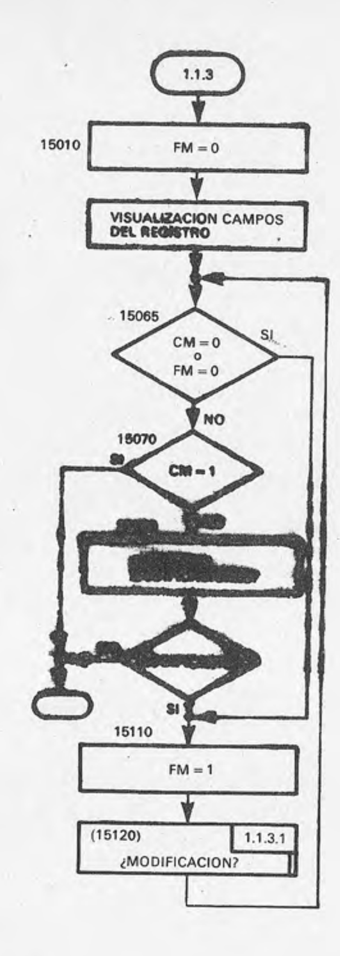
Borrado de una cadena

Si la cadena es nula (hay un cero en el fichero LNK.ENTRY en correspondencia con el registro base objeto de examen) se terminará la subrutina contenida en la figura 8. En caso contrario se proseguirá la ejecución del programa indicando también aquí si se libera un disco lleno. Después de grabar en RA la antigua cabecera de la lista libre, a DL se le da el valor de la cabecera a borrar; se escribe cero en el fichero LNK.ENTRY en el registro N-ésimo (lista nula para el registro base N) y se escribe DL en el registro 0 de LNK.ENTRY.

Si RA es igual a cero, es decir, si no existía ninguna lista libre, hemos acabado el proceso; en caso contrario se recorre toda la cadena eliminada hasta encontrar el último registro de la misma en el cual (con puntero en LNK.PTR) se escribe RA, es decir, la antigua cabecera de la lista libre. Por consiguiente, lo que hemos hecho es anular la cadena de registro N-ésimo e introducirla en la cabecera para la lista libre.

Introducción de campos y visualización

En diversos puntos del programa es necesario efectuar la visualización de datos, tanto relativos al registro base como a los registros controlados de listas; además, si se quiere controlar la inserción y la modificación de las informaciones se tendría una proliferación notable de instrucciones que haría tediosa la programa-




```
15000 REM ENTRADA DE LOS CAMPOS
15010 FM=0:FOR I=1 TO NC(J)
15020 VTAB(2+J*3+I):PRINT SPC(9-LEN(LB$K(I,J)));LB$(I,J);
15050 GOSUB 15420
15060 NEXT
15065 IF NOT (CM OR FM)THEN 15110
15070 IF CM=1 THEN RETURN
15080 VTAB (4+J*3+NC(J)):HTAB 20:PRINT M$;:GET A$:PRINT
15100 IF A$<>"S" THEN RETURN
15110 FM=1;SL(J)=1
15120 FOR I=1 TO NC(J):IF J=0 AND CM=2 AND I=1 THEN I=2
15130 VTAB(3+J*3+I):HTAB 11:CALL -868:PRINT LEFT$(T1$,LL(I,J))
15140 GOSUB 15400
15150 VTAB(2+J*3+I):HTAB 11:INPUT "";A$
```



```
15160 IF A$="" THEN 15200
15170 IF TP(I,J)=2 THEN RV$(I,J)=LEFT$(STR$(VAL(A$)),LL(I,J))
15180 IF TP(I,J)=1 THEN RV$(I,J)=LEFT$(A$,LL(I,J))
15190 IF A$<>RV$(I,J) THEN PRINT CHR$(7):GOTO 15140
15200 GOSUB 15400
15210 VTAB (I+J*3+I):HTAB 11:CALL -868
15215 IF RV$(1,0)="" THEN 15120
15220 NEXT
15230 GOTO 15065
15400 REM VISUALIZACION
15410 VTAB(2+J*3+I):HTAB 10:CALL -868
15420 PRINT SPC(1+(LL(I,J)-LEN(RV$(I,J)))*(TP(I,J)=2));RV$(I,J)
15430 RETURN
```


Figura 9.—Introducción de los campos y visualización.

ción y disminuiría la disponibilidad de memoria. Se tiene, pues, la exigencia de tener que condensar en una sola rutina versátil todas las necesidades relativas a la visualización, inserción y modificación de campos correspondientes a los registros base y auxiliares.

El acceso a dicha subrutina se realiza con parámetros de definición de las funciones deseadas, según las claves para comando:

CM=0: inserción
CM=1: visualización
CM=2: modificación

y "J" como indicador del tipo de archivo en el que se trabaja:

J=0: registro de base de datos
J=1: registro controlado de lista.

Por supuesto, la subrutina hace referencia a los nombres de los campos almacenados en la matriz LB$(*J), al registro actual en la matriz RV$(*J), al tipo de campo TP(*J), a la longitud correspondiente LL(*J) y al número de campos NC(J).

En la figura 9 se pone de manifiesto cuál es el efecto de la variable de control CM. Procediendo de forma ordenada se pone a cero el indicador de modificación y se visualizan los campos; si se trata de una sola visualización (CM=1) se vuelve a la subrutina que efectuó la llamada.

Para la inserción (CM=1) o la solicitud de modificación (CM=2) se pone FL=1 y se pasa a la modificación según se indica en la figura 10.

No se permite cambiar la clave principal en la fase de modificación, puesto que si se hiciera así se plantearían problemas de ordenación de registro.

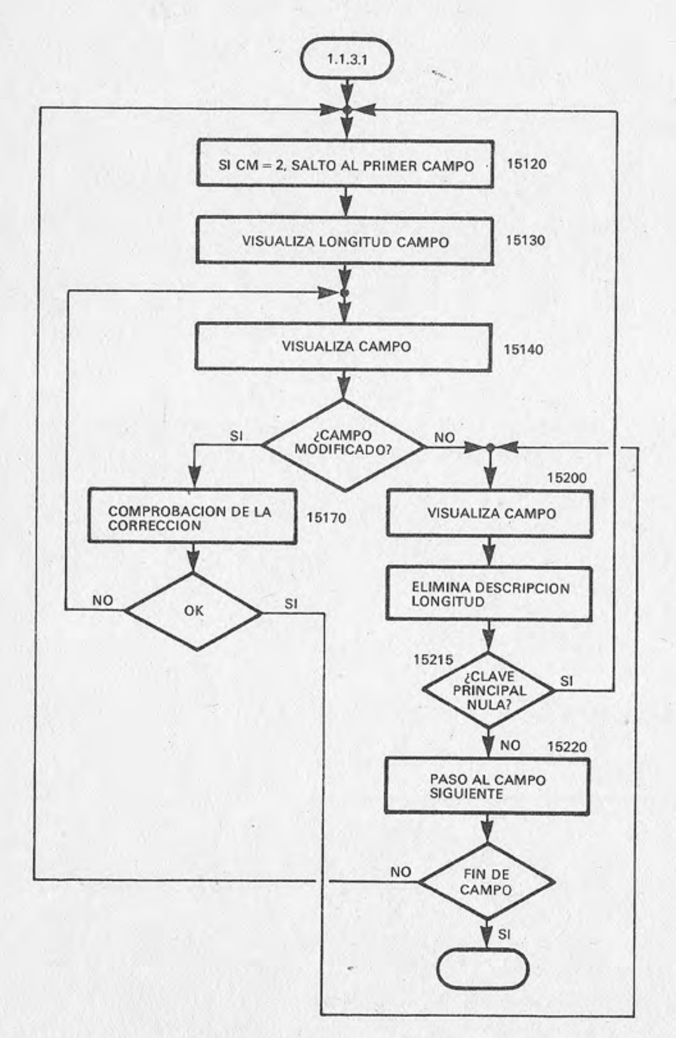


Figura 10.—Modificaciones de los campos (para el listado, véase figura 9).

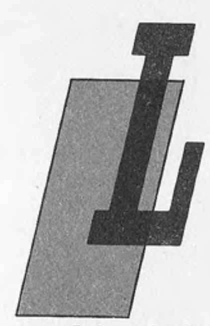
Para cualquier campo, la modificación se efectúa proporcionando una indicación de la longitud máxima admitida.

Además, cada campo puede ser confirmado con RETURN o modificado tanto de forma total como parcial; en tal caso, se realiza la comprobación de la corrección y, antes de pasar al campo siguiente se visualiza la etapa final y se elimina el indicador de longitud. Además, se comprueba que no es nula la clave principal.

Conclusiones

Ahora disponemos de un instrumento muy completo que le permite una gestión sofisticada de sus datos. Por consiguiente, puede comenzar a practicar con la manipulación de los archivos que la base de datos personal pone a su disposición (¡si tiene la paciencia de teclear las instrucciones!).

APENDICE



La creatividad y la invención suelen ser fundamentales en el perfeccionamiento, tanto estético como práctico, de un programa.

Una base de datos, tal como la presentada en esta monografía puede estar sujeta pues a modificaciones, incluso importantes, en la búsqueda de un instrumento que permita facilitar el propio trabajo o lograr los fines para los que se utiliza el programa.

Además de las líneas de programa no indicadas en las diversas figuras, incluimos también una subrutina para la impresión de etiquetas y otras pequeñas modificaciones puestas de manifiesto en el listado completo del programa. Asimismo se proporciona un tercer programa (Chequeo), llamado en el punto 9 del menú de la base de datos, concebido para la verificación y control de los propios archivos.

En resumen, se dan algunos ejemplos que reafirman la gran flexibilidad y adaptación de este programa para satisfacer las propias necesidades y que, gracias a la dosis de ingenio que sin duda poseen los lectores, les guiarán en la realización de una base de datos todavía más "personal".

Programa principal

```
1 PRINT "MAXFILES 3"
10 GOTO 60000
100 REM ****************************
101 REM ***    A P P L E  I I     **
102 REM **                        **
104 REM ** BASE DE DATOS PERSONAL **
105 REM ****************************
600 REM BUSQUEDA BINARIA
610 F = 0: L = 1: N = NX: LK = LEN (KM$)
620 IF FI THEN N = IV:IF LK > IA(FI,1) THEN LK = IA(FI,1)
630 IF N < L THEN RETURN : REM NO ENCONTRADO
640 P = INT ((L + N) / 2):PRINT R$(XF);P:INPUT A$:PRINT D$
650 IF LEFT$ (KM$,LK) < LEFT$ (A$,LK) THEN N = P - 1:GOTO 630
660 IF LEFT$ (KM$,LK) > LEFT$ (A$,LK) THEN L = P + 1:GOTO 630
670 F = 1:RETURN: REM ENCONTRADO
700 REM LECTURA CAMPOS DE BASE DE DATOS
710 PRINT R$(XF);P:INPUT IX$:INPUT IX%
720 PRINT R$(1);IX%:J=0
800 REM LECTURA DE CAMPOS
810 FOR I=1 TO N(J):INPUT RV$(I,J):NEXT
820 PRINT D$
830 RETURN
900 REM NO ENCONTRADO
910 IF F THEN  RETURN
920 VTAB 3:HTAB 15:PRINT "NO HAY"
930 VTAB 8:HTAB 22:GOSUB 58000
940 POP:RETURN
1000 REM LECTURA DEL FICHERO SIST.CTRL
1010 B$(0)="":B$(1)=",R"
1020 PRINT O1$"SIST.CTRL,S6,D1"
1030 PRINT R1$"SIST.CTRL": INPUT NC(0),EM
1040 FOR I=0 TO 9:INPUT E(I):NEXT :LT=E(0)
1050 FOR I=1 TO NC(0):INPUT LB$(I,0),TP(I,0),LL(I,0):NEXT
1060 FOR I=0 TO 10
1070 L="":INPUT A$,R(I),S(I),D(I):IF R(I) THEN L$= ",L"=STR$(R(I))
1080 O$(I)=O1$+A$+L$+",S"+STR$(S(I))+",D"+STR$(D(I))
1090 R$(I)=R1$+A$+B$+(R(I))0)
1100 W$(I)=W1$+A$+B$(R(I))0)
1110 C$(I)=C1$+A$
1120 NEXT
1130 PRINT C1$"SIST.CRL"
1140 RETURN
1500 REM LECTURA DEL FICHERO BD.CTRL
1510 PRINT O$(2)
1520 PRINT R$(2):NX,EA,PL,PT,CN,DF
1530 FOR I=1 TO 4:INPUT IA(I,0),IA(I,1),IA(I,2):NEXT
```



```
1570 PRINT C$(2)
1580 RETURN
2000 REM ESCRITURA DEL FICHERO BD.CTRL
2010 PRINT O$(2)
2020 PRINT W$(2):PRINT NX","EA","PL","PT","CN","DF
2040 FOR I=1 TO 4:PRINT IA(I,0)","IA(I,1)","IA(I,2):NEXT
2070 PRINT C$ (2)
2080 RETURN
2500 REM INSERCION DEL VECTOR ENTRY EN EL VECTOR INDEX
2505 VTAB 23:HTAB 1:PRINT "ORDENACION EN CURSO. ATENCION!"
2510 I=NX:NX=NX+K:IF EA < NX THEN EA=NX
2520 FOR K=K TO 1 STEP -1
2530 FOR I=I TO 1 STEP -1
2535 PRINT R$(0)I:INPUT IX$:INPUT IX%:PRINT W$(0)(I+K)
2540 IF EY$(K) < IX$ THEN PRINT IX$:PRINT IX%:NEXT
2550 PRINT EY$:PRINT EY%(K)
2560 NEXT K
2570 PRINT C$(0):PRINT O$(0)
2580 RETURN
3100 REM INSERCION DE UNA NUEVA CLAVE EN EL VECTOR ENTRY
3105 A%=EY%(K)
3110 FOR I=K-1 TO 1 STEP -1
3120 IF EY$(I) > RV$(1,0) THEN EY$(I+1) = EY$(I):EY%(I+1)=EY%(I):NEXT
3130 EY$(I+1)=RV$(1,0):EY%(I+1)=A%
3140 RETURN
3500 REM BUSQUEDA HACIA ATRAS
3510 F=D
3520 FOR P=P TO 1 STEP -1
3540 GOSUB 3600
3550 IF KM$=LEFT$(A$, LEN,(KM$)) THEN  F=1:NEXT
3560 P=P+1
3570 RETURN
3600 REM LECTURA DE INDICE EN DISCO
3610 PRINT R$(XF);P:INPUT A$:INPUT A%
3620 IF FI THEN IF LEN(KM$) > IA(FI,1) THEN PRINT R$(1);A%:FOR I=1 TO IA(FI,0):I
NPUT A$:NEXT
3630 PRINT O$
3640 RETURN
3800 REM LECTURA EN DISCO DE LA CLAVE DE ACCESO A PANTALLA
3810 P=P+1
3820 PRINT R$(XF);P:INPUT A$:INPUT A%:IF NOT FI THEN  3850
3830 PRINT R$(1);A$:INPUT KO$
3840 IF IA(FI,1) < LL(IA(FI,0)0) THEN FOR HK=2 TO IA(FI,0):INPUT A$:NEXT
3850 PRINT D$
3860 P=P-1
3870 RETURN
4000 REM BUSQUEDA POR INDICES
4010 B$="SELECCIONAR POR NUMERO":E$="RETURN PARA "
4020 GOSUB 600:GOSUB 900
```

```
4030 GOSUB 3500:GOSUB 900
4040 IF (P=EM) OR (FI AND P=IV) THEN RETURN
4050 I=1:GOSUB 3800
4070 IF KM$<> LEFT$ (A$,LEN(KM$)) THEN RETURN
4080 J=NX:IF FI THEN J=IV
4090 PRINT
4100 FOR I=0 TO J-P
4110 GOSUB 3800
4120 IF KM$<>LEFT$(A$,LEN(KM$)) THEN GOSUB 4300
4130 PRINT SPC(3-LEN(STR$(1+1)));I+1" "A$;
4134 IF FI THEN  HTAB (39-LL(1,0):PRINT " "KO$;
4136 PRINT
4140 IF I=INT (I/17)*17 AND I THEN GOSUB 4400
4150 NEXT
4160 GOSUB 4300:RETURN
4300 REM SELECCION FINAL
4310 VTAB 22:PRINT B$
4320 PRINT E$;:INPUT "SALIR";A$:A%=VAL(A$)
4330 IF NOT A% THEN F=0:GOTO 4360
4340 IF A%<1 OR A%>I THEN 4310
4350 P=P+A%-1
4360 POP:RETURN
4400 REM SELECCION MEDIA
4410 VTAB 22:PRINT B$
4420 PRINT E$;:INPUT "CONTINUAR";A$:A%=VAL(A$)
4430 IF NOT A% THEN VTAB 3:CALL -958:RETURN
4440 IF A%<1 OR A%>I+1 THEN 4410
4450 P=P+A%-1
4460 POP:RETURN
9000 REM BUSQUEDA VARIACIONES DE LOS CAMPOS INDICES
9010 FOR I=1 TO 4
9020 IF IA(I,0) THEN IF RV$(IA(I,0),0) <> RB$(IA(I,0),0) THEN IA(I,2)=0
9030 NEXT
9040 RETURN
10000 REM LECTURA DEL FICHERO LINK CTRL (CONTROL CADENA)
10010 PRINT O$(3)
10020 PRINT R$(3):INPUT NC(1)
10030 FOR I=1 TO NC(1)
10040 INPUT LB$(I,1),TP(I,1),LL(I,1)
10050 NEXT
10060 PRINT C$(3)
10070 FOR I=4 TO 6:PRINT O$(I):NEXT
10080 PRINT R$(4);0:INPUT DL
10090 PRINT R$(5);0:INPUT IL
10100 PRINT R$(5);INPUT IL
10110 PRINT D$
10120 RETURN
10500 REM LECTURA INDICES
10550 PRINT O$(0)
```



```
10560 PRINT O$(1)
10570 RETURN
15000 REM ENTRADA DE LOS CAMPOS
15010 FM=0:FOR  I=1 TO NC(J)
15020 VTAB(2+J*3+I):PRINT SPC(9-LEN(LB$K(I,J)));LB$(I,J);
15050 GOSUB 15420
15060 NEXT
15065 IF NOT (CM OR FM)THEN 15110
15070 IF CM=1 THEN RETURN
15080 VTAB (4+J*3+NC(J)):HTAB 20:PRINT M$;:GET A$:PRINT
15100 IF A$<>"S" THEN  RETURN
15110 FM=1;SL(J)=1
15120 FOR I=1 TO NC(J):IF J=0 AND CM=2 AND I=1 THEN  I=2
15130 VTAB(3+J*3+I):HTAB 11:CALL -868:PRINT  LEFT$(T!$,LL(I,J))
15140 GOSUB 15400
15150 VTAB(2+J*3+I):HTAB 11:INPUT "";A$
15160 IF A$="" THEN 15200
15170 IF TP(I,J)=2 THEN RV$(I,J)=LEFT$(STR$(VAL(A$)),LL(I,J))
15180 IF TP(I,J)=1 THEN RV$(I,J)=LEFT$(A$,LL(I,J))
15190 IF A$<>RV$(I,J) THEN PRINT CHR$(7):GOTO 15140
15200 GOSUB 15400
15210 VTAB (I+J*3+I):HTAB 11:CALL -868
15215 IF RV$(1,0)="" THEN 15120
15220 NEXT
15230 GOTO 15065
15400 REM VISUALIZACION
15410 VTAB(2+J*3+I):HTAB 10:CALL -868
15420 PRINT SPC(1+(LL(I,J)-LEN(RV$(I,J)))*(TP(I,J)=2));RV$(I,J)
15430 RETURN
20000 REM INSERCION DEL REGISTRO EN EL ARCHIVO BASE
20010 IF NX=EM THEN  GOSUB 59000:RETURN
20020 FOR  I=1 TO 4:IA(1,2)=0:NEXT
20050 GOSUB 24000
20060 K=1
20070 CM=0:J=0
20080 HOME:HTAB 15:PRINT "INSERCION"
20090 FOR I=1 TO NC(0):RV$(I,0)="":NEXT
20120 GOSUB 15000
20150 PRINT W$(1);EY%(K)
20160 FOR I= 1 TO NC(0):PRINT RV$(I,0):NEXT
20170 PRINT D$
20200 GOSUB 3100
20210 VTAB 23:CALL -868:PRINT "OTROS REGISTROS?";:GET A$:PRINT :IF A$="N" THEN 2
0260
20230 IF NX+K=EM THEN GOSUB 59000:GOTO 20260
20240 IF K=25 THEN GOSUB 2500:GOSUB 2000:GOSUB 24000:K=0
20250 K=K+1:GOTO 20080
20260 GOSUB 2500:GOSUB 2000
20270 PRINT C$(1):PRINT O$(1)
```

```
20280 RETURN
24000 REM INICIALIZACION ENTRADA
24005 EY$(0)=""
24010 FOR I=1 TO 25
24020 EY$(I)=FR$:EY%(I)=NX+I
24030 IF EA >= NX+I THEN PRINT R$(0)(NX+I):INPUT A$:INPUT EY%(I):PRINT D$
24040 RA=DL:DL=RO
24050 RETURN
25000 REM ELIMINACION
25010 IF FI THEN PRINT C$(XF):FI=0
25020 C=0
25030 FOR I=0 TO 20:EY$(I)=FP$:EY%(I)=0:NEXT
25040 CA=1:GOSUB 30000
25050 IF KM$="" THEN 25200
25060 CM=1:J=0:HOME:GOSUB 15000
25070 VTAB 22:INPUT "LO CONFIRMA? ";B$
25080 IF B$<> "S" THEN 25040
25090 GOSUB 26000
25100 IF C=19 THEN GOSUB 25500:GOTO 25020
25130 GOTO 25040
25200 GOSUB 25500
25210 PRINT C$(0)
25230 PRINT D$(0)
25240 IF FL THEN FOR I=4 TO 6:PRINT C$(I):PRINT D$(I):NEXT
25280 EY$(0)="":CA=0
25290 RETURN
25500 REM ELIMINACION FISICA
25505 IF NOT C THEN RETURN
25510 HOME:HTAB 10:PRINT "REGISTRO EN ELIMINACION":PRINT
25520 FOR I=0 TO C-1:PRINT EY$(I):NEXT
25530 PRINT :INPUT "LO CONFIRMA? ";A$
25540 IF A$<>"S" THEN  RETURN
25560 FOR I=0 TO C-1
25570 N=EY%(I)
25580 IF FL THEN GOSUB 29000
25590 NEXT
25600 GOSUB 26250
25620 GOSUB 2000
25630 RETURN
26000 REM INSERCION DEL REGISTRO ELIMINADO EN EL VECTOR ENTRY
26010 FOR I=0 TO C
26020 IF EY$(I) < IX$ THEN NEXT
26030 IF EY%(I) = IX% THEN RETURN
26040 FOR N=18 TO I STEP -1
26050 EY$(N+1)=EY$(N):EY%(N+1)=EY%(N)
26060 NEXT
26070 EY$(I)=IX%:EY%(I)=IX%
26080 C=C+1:RETURN
26250 REM COMPACTACION INDICES ANTIGUOS (ACTUALIZACION DEL INDICE)
```



```
26255 PRINT :PRINT "COMPACTACION: ";:FLASH:PRINT "ATENCION":NORMAL
26260 K=0:KM$=EY(0):GOSUB 600:GOSUB 3500
26270 PRINT R$(0)I:INPUT IX$:INPUT IX%
26280 IF IX%=EY%(K) THEN K=K+1:GOTO 26300
26290 PRINT W$(0)(I-K):PRINT IX$:PRINT IX%
26300 NEXT
26310 FOR I=EA-K+1 TO EA:PRINT W$(0)I:PRINT FR$:PRINT EY%(I-EA+K-1):NEXT
26320 NX=NX-C:FOR I=1 TO 4:IA(I,2)=0:NEXT
26330 RETURN
29000 REM ELIMINACION EN CADENA
29010 PRINT R$(4);N:INPUT RO
29030 IF NOT RO THEN PRINT D$:RETURN
29035 SL(1)=1:IF DF THEN  DF=0:GOSUB 2000
29045 PRINT W$(4);N:PRINT 0
29050 PRINT W$(4);O:PRINT DL
29070 IF (NOT RA) THEN PRINT D$: RETURN
29080 PRINT R$(5);RO:INPUT A
29100 IF A<>0 THEN RO=A:GOTO 29080
29110 PRINT W$(5);RO:PRINT RA:PRINT D$
29120 RETURN
30000 REM BUSQUEDA Y MODIFICACION
30010 HOME
30020 PRINT LB$(1+(FI)0)$(IA(FI,0);:INPUT ": ";KM$
30030 IF KM$="" AND (NOT CA) THEN 30250
30035 IF KM$="" AND (CA) THEN RETURN
30040 GOSUB 4000:IF NOT F THEN  30000
30060 GOSUB 700
30100 FOR I=1 TO NC(J):RB$(I,J):NEXT
30110 IF (CA) THEN  RETURN
30120 CM=2-FL:HOME:GOSUB 15000
30140 IF NOT FM THEN  30210
30150 GOSUB 9000
30160 PRINT W$(1);IX%
30170 FOR I=1 TO NC(0):PRINT RV$(I,0):NEXT :PRINT D$
30210 IF FL THEN N=IX%:GOSUB 35000
30220 GOTO 30000
30250 IF SL(0) THEN PRINT C$(1):PRINT D$(1):GOSUB 2000
30270 IF SL(1) THEN FOR I=4 TO 6:PRINT C$(I):PRINT D$(I):NEXT
30290 RETURN
35000 REM GESTION DE LAS CADENAS
35010 VTAB 19:CALL -958:PRINT " 1- INSERCION EN CADENA"
35020 PRINT " 2- EXPLORACION Y ELIMINACION"
35025 PRINT " 3- ELIMINACIN CADENA"
35030 INPUT "(RETURN PARA SALIR) ";A$
35040 IF A$="" THEN  RETURN
35050 ON VAL(A$) GOSUB 36000,37000,29000
35060 GOTO 35000
36000 REM INSERCION DEL REGISTRO EN CADENA
36010 IF DF THEN PRINT "DISCO LLENO";:GOSUB 58000:RETURN
```

```
36020 ONERR GOTO 38100
36030 VTAB 5:CALL -958
36040 J=1:CM=0:FOR I=1 TO NC(1):RV$(I,1)="":NEXT :GOSUB 15000
36050 X=DL:IF NOT X THEN X=IL
36060 PRINT W$(6);X
36070 FOR I=1 TO NC(1):PRINT RV$(I,1):NEXT
36080 IF NOT DL THEN 36140
36090 PRINT R$(5);X:INPUT DL
36110 PRINT W$(4);0:PRINT DL
36130 GOTO 36170
36140 IL=IL+1
36150 PRINT W$(5);0:PRINT IL
36170 PRINT R$(4);N:INPUT A
36185 ONERR GOTO 38000
36190 PRINT W$(5);X:PRINT A
36210 PRINT W$(4);N:PRINT X
36230 PRINT D$:POKE 216,0
36240 RETURN
37000 REM EXPLORACION DE LA CADENA DE REGISTROS CON POSIBILIDAD DE ELIMINACION
37010 RD=0:FD=0
37020 PRINT R$(4);N
37030 INPUT RA:PRINT D$
37040 IF(NOT RA) THEN VTAB 22:CALL -958:PRINT "FIN ":GOTO 58000
37050 PRINT R$(6);RA
37060 J=1:GOSUB 800
37090 CM=2:VTAB 5:CALL -958
37100 GOSUB 15000
37110 IF NOT FM THEN 37140
37120 PRINT W$(6);RA
37130 FOR I=1 TO NC(1):PRINT RV$(I,1):NEXT
37140 PRINT D$:VTAB 20:CALL -958:PRINT "E)XIT    D)EL."
37150 PRINT "(RETURN PARA CONTINUAR)";:GET A$:PRINT
37160 IF A$="E" THEN RETURN
37170 IF A$="D" AND NOT FD THEN 37250
37180 IF A$<> CR$ THEN 37140
37190 RD=RA:FD=0
37200 PRINT R$(5);RO
37210 GOTO 37030
37250 REM ELIMINACION CADENA
37260 FD=1:SL(1)=1
37270 IF DF THEN DF=0:GOSUB 2000
37280 PRINT R$(5);RA:INPUT A
37290 IF NOT RD THEN  37310
37300 PRINT W$(5);RO:GOTO 37320
37310 PRINT W$(4);N
37320 PRINT A
37330 PRINT W$(5);RA
37340 PRINT DL
37350 DL=RA
```



```
37360 PRINT W$(4);0
37370 PRINT DL
37380 VTAB 5:CALL -958:RA=RO
37390 PRINT D$
37400 IF RD THEN 37040
37410 RETURN
38000 REM DISCO LLENO
38010 IL=IL-1
38020 PRINT O$(5)
38030 PRINT W$(5);0:PRINT IL
38040 PRINT C$(5)
38100 HOME:HTAB 15:PRINT "DISCO LLENO":PRINT :GOSUB 58000
38140 DF=1 :GOSUB 2000
38150 POKE 216,0
38160 RUN
40000 REM LISTADO
40010 HOME:VTAB 9:HTAB 9:PRINT M$:GET A$
40030 IF A$="S" THEN GOSUB 47000:GOTO 40000
40040 HOME:VTAB 9:HTAB 9:PRINT "IMPRESION? ";: GET A$:PRINT :HOME
40050 CL=40:S=0:IF A$"S" THEN CL=CN:S=1
40060 GOSUB 49800
40100 NP=1+(FI>0)*(IA(FI,0)-1)
40110 P=1:KM$=MI$(NP,0):IF KM$<>"" THEN GOSUB 600:IF F THEN GOSUB 3500
40120 SL=P
40140 KM$=(NP,0):GOSUB 600:IF F THEN  GOSUB 3500
40150 EL=P:NP=1
40200 IF S THEN PRINT D$"PRM1":IF NOT PT THEN PRINT CI;CN;"N"
40300 FOR P=SL TO EL:IF PEEK(-16384)=155 THEN P=EL:A$=EC$:GOTO 40370
40310 GOSUB 700:IF SL(J) THEN GOSUB 42500:IF NOT SE THEN 40370
40315 IF SC AND FL THEN GOSUB 40900:IF NOT SE THEN 40370
40320 GOSUB 41000
40330 GOSUB 43000
40340 GOSUB 42000
40350 GOSUB 41500
40360 IF FL THEN GOSUB 40600
40370 NEXT
40400 IF NOT (V%(0,0) OR V%(0,1) THEN  40500
40410 IF FL THEN T(0)=1:GOSUB 42800:L=1
40420 IF T(0) THEN HOME
40430 IF NP THEN L=1
40450 J=0:GOSUB 43500:J=1:OFR I=1 TO NC(J)+:TT(I,J)=TT(I,2):NEXT
40460 IF V%(0,1) AND FL THEN GOSUB 41500:T(J) =1 :HOME:GOSUB 43500
40500 GOSUB 42800
40510 PRINT D$"PRMO":IF A$<>EC$ AND NOT S THEN  VTAB 23:GOSUB 58000
40520 POKE -1638,0
40530 RETURN
40600 REM LISTADO CADENA
40610 J=1:PRINT R$(4);IX%:INPUT K:IF NOT K THEN  RETURN
40620 SE=1:PRINT R$(6);K:GOSUB 800:IF SL(J) THEN GOSUB 42500
```

```
40625 IF SE AND NP THEN  40700
40626 IF SE THEN NP=1:ON 2*S+T(J)+1 GOSUB 41300,41400,42300,42900:GOTO 40710
40630 PRINT R$(5);K:INPUT K:PRINT D$:IF PEEK (-16384) <> 155 AND K THEN 40620
40640 RETURN
40700 GOSUB 41000
40710 GOSUB 43000
40720 GOSUB 42000
40730 GOSUB 41500:IF P>EL THEN  40800
40740 PRINT R$(5);K:INPUT K:PRINT D$:IF NOT K THEN 40800
40745 IF PEEK(-16384)=155 THEN A$=EC$:GOTO 40800
40750 PRINT R$(6);K:GOSUB 800:IF SL(J) THEN GOSUB 42500:IF NOT SE THEN 40740
40760 GOTO 40700
40800 GOSUB 41000:GOSUB 43500:IF (NOT T(1)) AND (A$<>EC$) AND (NOT S) THEN GOSUB
      42700
40810 GOSUB 42800:NP=1:L=1
40820 RETURN
40900 REM PRESELECCION
40910 SE=0:PRINT R$(4)IX%:INPUT K:PRINT D$:IF NOT K THEN RETURN
40920 IF NOT SL(1) THEN SE=1:RETURN
40930 J=1:SE=1:PRINT R$(6);K:GOSUB 800:GOSUB 42500
40940 IF SE THEN J=0:RETURN
40950 PRINT R$(5);K:INPUT K:PRINT D$:IF PEEK(-1638)<>155 AND K THEN 40930
40960 SE=0:J=0
40999 RETURN
41000 REM CABECERA
41010 ON 4*J+2*S+T(J) GOTO 41200,41100,41200,41300,41400,41100,41200
41100 REM DB-V-O
41110 IF NP THEN HOME:L=1:GOSUB 42300
41120 RETURN
41200 REM DB-V-V
41210 IF NP THEN L=1
41220 HOME:GOSUB 42900:RETURN
41300 REM CADENA-V-O
41310 IF NP THEN VTAB 5:CALL -958:L=5:GOSUB 42300
41320 RETURN
41400 REM CADENA-V-V
41410 VTAB 5:CALL -958:GOSUB 42900:L=6:RETURN
41500 REM FIN DE PAGINA
41510 NP=0:IF L=1 THEN RETURN
41520 RA=20+(LP-25-(NC(J)+O%(0,J))*T(J)*S
41530 ON 4*J+2*S+T(J) GOTO 41610,41700,41700,41600,41610,41700,41700
41600 IF RA>L THEN RETURN
41610 GOSUB 42700:NP=1:RETURN
41700 IF RA<L THEN GOSUB 42800:NP=1
41710 RETURN
42000 REM SUBLISTADO
42010 ON T(J)+1 GOSUB 42100,42200
42020 IF PEEK(-16384)=155 THEN P=EL+1:GET A$
42030 B$="":IF T(J) THEN B$=""
```



```
42040 PRINT B$:L=L+1
42090 RETURN
42100 REM HORIZONTAL
42110 PRINT SPC((CL-L%(0,J))/2);
42120 FOR I=1 TO NC(J) +O%(0,J): IF %(I,J) THEN NEXT: RETURN
42130 C=L%(I,J) - LEN (RV$(I,J))
42140 A=(I>1): IF TP(I,J)=2 THEN A=A +C:C=0
42150 PRINT SPC(A) RV$(I,J)SPC(C);
42160 NEXT
42170 RETURN
42200 REM VERTICAL
42210 VTAB 3+3*J
42220 FOR I=1 TO NC(J) + O%(0,J): IF I%(I,J) THEN  NEXT: RETURN
42240 PRINT SPC(9-LEN(LB$(I,J)));LB$(I,J);
42250 PRINT SPC(1+(24-LEN (RV$(I,J)))*(TP(I,J) = 2));RV$(I,J)
42260 L=L+1
42270 NEXT
42280 RETURN
42300 REM CABECERA
42310 FOR I=1 TO NC(J)+1:RB$(I,J)=RV$(I,J):RV$(I,J) = RB$(I,J): NEXT
42340 RETURN
42500 REM SELECCION
42510 SE=0: FOR I=1 TO NC(J)
42520 IF TP(I,J) = 1 THEN IF RV$(I,J)<MI$(I,J) OR RV$(I,J)> MA$(I,J) THEN RETURN

42530 IF TP(I,J) = 2 THEN  A=VAL (RV$(I,J)): C=VAL(MA$(I,J)):IF A<VAL(MI$(I,J))
OR A>C+1E+37*(NOT C )) THEN RETURN
42540 NEXT : SE = 1
42550 RETURN
42700 REM ESPERA TECLADO
42710 IF A <> EC$ THEN VTAB 23: PRINT "ESC O RET? ";: GET A$: PRINT: VTAB 20
42720 IF A$ = EC$ THEN P=EL+1
42730 RETURN
42800 REM AVANCE DE PAPEL
42810 REM
42820 REM
42900 REM AVANCE DE LINEA
42910 PRINT "":L=L+1
42930 RETURN
43000 REM CALC
43010 IF O$(0,J) THEN A=0: FOR I=1 TO NC(J): A=A+VAL(RV$(I,J))*O%(I,J): NEXT:RV$
(I,J) = STR$(A)
43020 IF V%(0,J) THEN  FOR I=1 TO NC(J)+O%(0,J):TT(I,J)=TT(I,J)+VAL(RV$(I,J))*V%
(I,J):NEXT
43030 RETURN
43500 REM LISTADO TOTAL
43505 IF NOT V%(0,J) THEN  RETURN
43507 FOR I=1 TO NC(J)+1:RV$(I,J)="":IF V%(I,J) THEN RV$(I,J)=LEFT$(T1$,L%(I,J))
```

```
43508 NEXT : IF NOT T(J) THEN GOSUB 42000
43510 FOR I=1 TO NC(J)+1
43520 IF V%(I,J) THEN RV$(I,J)=STR$(TT(I,J))
43530 NEXT
43540 GOSUB 42000
43550 IF J THEN FOR I=1 TO NC(J)+1:TT(I,2) = TT(I,2) + TT(I,J): TT(I,J) = 0: NEX
T
43560 RETURN
44000 REM CRITERIO ENTRADA
44010 FOR I = 1 TO NC(J)
44020 VTAB 4+I: HTAB 13:GET A$
44030 IF A$=EC$ THEN I%(I,J)=1: CALL -868:V%(I,J) I=0: O%(I,J)=0: GOTO 44120
44040 IF A$<>CR$ THEN I%(I,J)=0: PRINT "X";
44060 HTAB 23: GET A$
44070 IF A$=EC$ THEN V%(I,J)=0: PRINT "";: GOTO 44090
44080 IF A$<>CR$ THEN V%(I,J) = 1: PRINT "X";
44090 HTAB 34 : GET A$
44100 IF A$=EC$ THEN O$ (I,J) = 0: PRINT "" ;: GOTO 44120
44110 IF A$ <> CR$ THEN O$(I,J) = 1: PRINT "X";
44120 NEXT : PRINT
44130 RETURN
45000 REM CRITERIOS LISTA VISUALIZACION
45010 HOME : PRINT SPC(18);"FORMATO"
45030 PRINT :HTAB 20 : PRINT LB$(NM,J) SPC(5) LB$(NM,J)
45040 HTAB 11 : PRINT "LISTADO VERTICAL HORIZONTAL "
45045 GOSUB 49000
45050 FOR I=1 TO NC(J); IF I%(I,J) THEN 45090
45060 VTAB 4 + I: HTAB 13: PRINT "X";
45070 IF V%(I,J) THEN HTAB 23: PRINT "X";
45080 IF O%(I,J) THEN HTAB 34: PRINT "X";
45090 NEXT : PRINT
45100 RETURN
46000 REM CRITERIOS SELECCION ENTRADA
46010 FOR I = 1 TO NC(J)
46020 VTAB 4 + I : HTAB 11 : GET A$
46030 IF A$ = EC$ THEN MI$(I,J) = "": PRINT SPC(15): GOTO 46060
46040 IF A$<> CR$ THEN PRINT A$;:INPUT "";B$:MI$(I,J)=A$+B$
46060 VATB 4+I:HTAB 25:PRINT "";:GET A$
46070 IF A$=F$ THEN MA$(I,J)=FR$:CALL -868:GOTO 46100
46080 IF A$<>CR$ THEN PRINT A$;:INPUT "";B$:MA$(I,J)=A$+B$+FR$
46090 IF TP(I,J)=1 THEN IF MI$(I,J)>MA$(I,J) THEN 46020
46095 IF TP(I,J)=2 THEN IF VAL(MI$(I,J))>VAL(MA$(I,J) THEN 46020
46100 NEXT
46110 RETURN
47000 REM SELECCION PRINCIPAL
47005 SC=0
47010 FOR S=1 TO 2
47020 FOR J=0 TO FL
47030 ON S GOSUB 48000,45000
```



```
47040 VTAB 21:HTABE 20:PRINT M$;:GET A$
47050 IF A$="S" THEN ON SGOSUB 46000,44000!:GOTO 47030
47060 IF A$<>"N" THEN 47040
47062 IF NOT J AND S=1 AND FL THEN VTAB 23:HTAB 4:PRINT "SOLAMENTE LOS REGISTROS
 CON CADENA? ";:GET A$:PRINT
47064 IF A$<>"N" AND A$<>"S" THEN  47062
47066 IF A$="S" THEN SC=1
47070 NEXT J,S
47080 RETURN
48000 REM VISUALIZACION SELECCIONADA
48010 HOME:PRINT SPC(20)"SELECCION"
48015 HTAB 17:PRINT "DE"SPC(14)"A"
48020 GOSUB 49000
48030 FOR I=-1 TO NC(J)
48040 VTAB 4+I:HATB 11:IF MI$(I,J)<>"" THEN PRINT MI$(I,J)
48060 VTAB 4+I:HATB 26:IF MA$(I,J)<>FR$ THEN PRINT MA$(I,J)
48080 NEXT
48090 RETURN
49000 REM VISUALIZACION CAMPOS
49010 VTAB 4+I:PRINT SPC(9-LEN(LB$(I,J)))LB$(I,J)
49020 NEXT :VTAB 4+I:PRINT SPC(9-LEN(LB$(I,J)))LB$(I,J)
49030 NEXT
49040 RETURN
49800 REM INICIALIZACION  LISTA VARIABLE
49810 FOR J=0 TO FL:SL(J)=0:A=0:V%(0,J)=0:O%(0,J)=0
49820 FOR I=1 TO NC(J)
49830 IF MI$(I,J)<>"" OR MA$(I,J)<>FR$ THEN SL(J)=1
49840 IF I%(I,J) THEN 49890
49850 L%(I,J)=LL(I,J)<LEN(LB$(I,J)) THEN L%(I,J)=LEN(LB$(I,J))
49860 A=A+L%(I,J)+1
49870 IF V%(I,J) THEN V%(0,J)=1
49880 IF O%(I,J) THEN O%(I,J)=1
49890 NEXT
49900 LB$(I,J)="TOTAL":TP(I,J)=2:LL(I,J)=10
49910 L%(I,J)=LL(I,J):L%(0,J)=A+(1+L%(I,J))*O%(0,J):V%(I,J)=V%(0,J)
49915 T(J)=(CL<L%(0,J))
49920 NEXT :IF FL AND NOT S THEN T(0)=1
49930 NC(2)=NC(1):FOR J=0 TO FL*2:FOR I=1 TO NC(J)+1:TT(I,J)=0:NEXT I,J
49950 RETURN
50000 REM ETIQUETAS
50010 E=0:FOR I=1 TO 4:IF E(I) THEN E=E+1
50020 NEXT :HOME:VTAB 9:J=0
50040 IF E<3 THEN PRINT "DEFINIR FORMATO ETIQUETA":FOR I=1 TO 999:NEXT :RETURN
50045 PRINT "MODIFICACION DEL TIPO DE SELECCION? ";:GET A$
50050 IF A$="S" THEN GOSUB 50900:GOTO 5000
50060 PRINT :PRINT :PRINT "IMPRESION ETIQUETAS INDIVIDUALES? ";:GET A$
50070 OL=INT(CN/E(7)):IF A$="S" THEN OL=1
50080 PRINT :PRINT :PRINT "IMPRESORA PREPARADA? ";:GET A$:PRINT
50090 IF A$="N" THEN RETURN
```

```
50100 S=1":CL=CN:GOSUB 49800:HOME
50110 NP=1+(FI>0)*(IA(FI,0)-1)
50120 P=1:KM$=MI$(NP,0):IF KM$<>"" THEN GOSUB 600:IF F THEN GOSUB 3500
50130 SL=P
50140 KM$=MA$(NP,0):GOSUB 600:IF F THEN GOSUB 3500
50150 EL=P:NP=1:E=0
50200 PRINT D$"PR#1":IF NOT PT THEN PRINT CI$;CN"N"
50300 GOSUB 700:IF SL(J) THEN GOSUB 42500 :IF NOT SE THEN 50400
50320 E=E+1:RV$(0,0)=""
50330 FOR A=1 TO 4:ET$(A,E)=LEFT$(RV$(E(A),),E(7)-E(9)):NEXT
50340 ET$(4,E)=LEFT$(ET$(4,E),5)+""+LEFT$(RV$(E(A),0),E(7)-E(9)-6)
50350 IF E=DL THEN GOSUB 50700
50400 NEXT
50410 GOSUB 50700
50420 PRINT S$"PR#0"
50490 RETURN
50700 REM IMPRESION
50710 IF NOT E THEN RETURN
50720 L=LP-E(8):ON (E(8) GOSUB 42900,42800
50730 FOR A=1 TO 4:FOR B=1 TO E
50750 PRINT SPC(E(9))ET$(A,B);:IF B<E THEN PRINT SPC(E(7)-E(9)-LEN(ET$(A,B)));
50760 NEXT :PRINT :NEXT
50770 IF E(6) >E(8)+4 THEN L=LP-E(6)+E(8)+4:ON ((E(6)-E(8)-4)>1)+1 GOSUB 42900,4
2800
50780 E=0:RETURN
50900 REM SEL
50910 GOSUB 48000
50920 VTAB 21:HTAB 20:PRINT M$;:GET A$
50930 IF A$="S" THEN GOSUB 46000:GOTO 50910
50940 IF A$<>"N" THEN 50920
50950 RETURN
51000 REM CAMBIO DE INDICE
51010 IF FI THEN PRINT C$(FI+6)
51030 HOME:HTAB 15:PRINT "INDICES POSIBLES":PRINT :PRINT "0 ";LB$(1,0)
51040 FOR I=1 TO 4
51050 IF NOT IA(I,0) THEN  51080
51060 PRINT I" ";LB$(IA(I,0),0);:IF NOT IA(I,2) THEN HTAB 20:PRINT "NO";
51070 HTAB 24:PRINT "ACTUALIZADO"
51080 NEXT
51090 VTAB 10:INPUT "CUAL? ";A$:A%=VAL(A$)
51100 IF NOT A% THEN FI=0:XF=FI:RETURN
51110 IF A% >4 OR IA(A%,0)=0 THEN 51090
51120 FI=A%:XF=FI+6
51130 PRINT D$(XF)
51140 PRINT R$;0:INPUT IV:PRINT D$
51170 RETURN
52000 REM GESTION DE INDICADORES DE ESTADO DE LAS CADENAS
52010 IF PL AND FL THEN FOR I=4 TO 6:PRINT C$(I):NEXT
52020 IF PL THEN FL=NOT FL
```



```
52030 IF FL THEN GOSUB 10000
52040 RETURN
58000 REM ESPERA TECLADO
58010 INPUT "PULSE RETURN ";A$
58020 RETURN
59000 REM OVERFLOW BD
59010 HOME:PRINT "NO HAY LUGAR PARA OTROS REGISTROS"
59020 GOSUB 58000
59030 RETURN
60000 REM PRUEBA PRIMERA VEZ (DEFINICIONES DE LA ASIGNACION DE LA BASE DE DATOS)

60010 GOSUB 62000
60020 ONERR GOTO 60400
60030 GOSUB 1000
60040 GOSUB 1500
60050 POKE 216,0
60070 GOSUB 10500
60100 REM MENU PRINCIPAL
60105 TEXT
60110 HOME:HTAB 11:PRINT "BASE DE DATOS PERSONAL"
60120 VTAB 3:PRINT "REGISTROS PRESENTES EN ARCHIVO";NX
60130 PRINT "MAXIMOS REGISTROS POSIBLES      ";EM
60140 VTAB 6:HTAB 10:PRINT "FASES DE ACTIVIDAD:"
60150 VTAB 8:PRINT " 1- INSERCION"
60160 PRINT " 2- ELIMINACION"
60170 PRINT " 3- BUSQUEDA Y/O MODIFICACION"
60180 PRINT " 4- LISTADO"
60190 PRINT " 5- ETIQUETA"
60200 PRINT " 6- CAMBIO DE INDICE"
60210 PRINT " 7-";:IF FL THEN PRINT "DES"
60215 PRINT " HABILITACION CADENAS"
60220 PRINT " 8- RECONFIGURACION DEL SISTEMA"
60230 PRINT " 9- VERIFICACION ARCHIVO"
60290 PRINT "10- FIN"
60295 VTAB 23:HTAB 10:PRINT "D & V PRODUCTION"
60300 VTAB 19:HTAB 10:PRINT "CUAL? ";A$:I=VAL(A$)
60320 IF (I>1 AND I<6) AND NOT NX THEN 60100
60330 ON I GOSUB 20000,25000,30000,40000,50000,51000,52000,60400,60450,60500
60340 SL(0)=0:SL(1)=0:GOTO 60100
60400 REM LANZAMIENTO PROGRAMA DE CONFIGURACION DEL SISTEMA
60410 PRINT D$"EJECUCION DE PROGRAMA DE CONFIGURACION DEL SISTEMA"
60450 PRINT D$"EJECUCION DEL PROGRAMA DE CHEQUEO"
60500 REM FINAL DEL PROGRAMA"
60510 PRINT C1$
60520 PRINT D$"LOCK LOGO,D1"
60530 PRINT D$"MAXFILES 3"
60540 HOME:PRINT "¡ ADIOS !"
60550 END
62000 REM INICIALIZACION VARIABLES
```

```
62010 I=0:T1$="-":FOR J=I TO 4:T1$=T1$+T1$:NEXT
62020 D$=CHR$(4):FR$=CHR$(95)
62030 NM=15
62040 DIM LB$(NM,1),TP(NM,1),LL(NM,1),RB$(NM,1),RV$(NM,1),EY$(25),EY%(25)
62050 O1$=D$+"OPEN":R1$=D$+"READ":W1$=D$+"WRITE":C1$=D$+"CLOSE"
62060 DIM MI$(NM,1),MA$(NM,1)
62070 FOR I=1 TO NM:FOR J=0 TO 1:MA$(I,J)=FR$:LB$(NM,J)="TOTAL":NEXT J,I
62080 DIM I%(NM,1),L%(NM,1),V%(NM,1),O%(NM,1),TT(NM,2),T(2),SL(1)
62090 CR$=CHR$(13):EC$=CHR$(27):CI$=(9)
62100 M$="MODIFICACIONES? "
62200 RETURN
```


Programa de configuración (SETUP)

```
1 REM **************************
2 REM *   A P P L E    I I     *
3 REM *                        *
4 REM * BASE DE DATOS PERSONAL *
5 REM **************************
10 GOTO 60000
500 REM HEAPSORT
505 FOR L=0 TO NX:PI%(L)=L:NEXT
510 IF NX=1 THEN RETURN
515 L=INT(NX/2)+1:R=NX
520 IF L>1 THEN L=L-1:PA=PI%(L):GOTO 540
525 PA=PI%(R):PI%(R)=PI%(1):R=R-1
530 IF R=1 THEN PI%(1)=PA:RETURN
540 J=L
545 I=J:J=2*J
550 IF J>R THEN PI%(I)=PA:GOTO 520
555 IF J<R THEN GOSUB 600
560 IF IX$(PA) < IX$(PI%(J)) THEN PI%(I)=PI%(J):GOTO 545
565 PI%(I)=PA:GOTO 520
600 II=P%(J):II=PI%(J+1):IF IX$(II)<IX$(II) THEN J=J+1:RETURN
610 IF IX$(II)>IX$(II) THEN RETURN
620 PRINT D$R$(1)IX%(II):FOR KK=1 TO NC:INPUT V1$(KK):NEXT
630 PRINT D$R$(1)IX%(II):FOR KK=1 TO NC:INPUT V2$(KK):NEXT:PRINT D$
640 II=VT(0):IF V1(II)<V2$(II) OR NOT NI THEN J=J+1:RETURN
650 II=VT(0):IF V1(II)>V2$(II) OR NOT NI THEN RETURN
660 FOR KK=1 TO NI:II=VT(KK)
670 IF TP(II)=1 THEN IF VI$(II)<V2$(II) THEN J=J+1:RETURN
680 IF TP(II)=2 THEN IF VAL(VI$(II))<VAL(V2$(II)) THEN J=J+1:RETURN
690 IF TP(II)=1 THEN IF VI$(II)=V2$(II) THEN NEXT
695 IF TP(II)=2 THEN IF VAL(VI$(II))=VAL(V2$(II)) THEN NEXT
699 NEXT
800 PRINT D$R$(1)IX%:FOR KK=1 TO NC:INPUT V1$(KK):NEXT
810 PRINT D$R$(1)IX%(PI%(J)):FOR KK=1 TO NC:INPUT V2$(KK):NEXT:PRINT D$
820 IF=VT(0):IF V1$(II)<V2$(II) THEN 890
830 IF (V1$(II)>IF V2$(II)) OR NOT NI THEN 565
840 FOR KK=1 TO NI:II=VT(KK)
850 IF TP(II)=1 THEN IF V1$(II)<V2$(II) THEN 885
860 IF TP(II)=2 THEN IF VAL(V1$(II))<VAL(V2$(II)) THEN 885
862 IF TP(II)=1 THEN IF V1$(II)=V2$(II) THEN NEXT
870 IF TP(II)=2 THEN IF VAL(V1$(II))=VAL(V2$(II)) THEN NEXT
880 GOTO 565
885 KK=NI:NEXT
890 PI%(I)=PI%(J):GOTO 545
3508 RETURN
15000 REM GESTION INDICES
15005 IF NOT NX THEN RETURN
```

```
15010 HOME:HTAB 15:PRINT "INDICES EXISTENTES":PRINT
15020 FDOR I=1 TO 4
15030 PRINT I"   "LB$(IA(I,0);:HATB 15:PRINT "S"S(I+6)" D"D(I+6);:IF NOT IA(I,2)
    THEN HTAB 22:PRINT "NO";
15040 PRINT "  ACTUALIZADO":NEXT
15060 VTAB 8:PRINT " 1-ELIMINACION"
15070 PRINT " 2-CREACION"
15080 PRINT " 3-FIN"
15100 PRINT : INPUT "CUAL? "; A$: A% = VAL(A$)
15110 IF A$<1 OR A%>2 THEN 15000
15120 ON A% GOSUB 15200,15300
15130 IF A%<>3 THEN 15000
15140 GOSUB 30000:GOSUB 40000:GOSUB 42000
15150 RETURN
15200 REM ELIMINACION INDICE
15210 PRINT :INPUT "CUAL ELIMINAR? ";A$:A=VAL(A$)
15220 IF A<1 OR A>4 THEN 15200
15230 IF NOT IA(A,0) THEN RETURN
15250 PRINT D$"DELETE ";MI$(O$(A+6),6,7);:RIGHT$(O$(A+6),6)
15260 IA(A,0)=0:IA(A,2)=0
15270 RETURN
15300 REM CREACION INDICE
15310 FOR P=1 TO 4:IF IA(P,0) THEN NEXT
15320 IF P>4 THEN  RETURN
15330 VTAB 14:INPUT "QUE CAMPO? ";A$:A=VAL(A$):CALL -958
15340 IF <2 OR A> NC OR TP(A)=2 THEN 15330
15350 VTAB 14:HTAB 18:PRINT LB$(A);HTAB 29:INPUT "LO CONFIRMA? ";A$
15355 IF LEFT$(A$,1)<> "S" THEN 15330
15360 IA(P,0)=A:VT(0)=A
15365 GOSUB 16000:PRINT :IF NOT OK THEN 15330
15370 IA(P,1)=INT((FRE(0)-2000)/EM):IF IA(P,1)>LL(A) THEN IA(P,1)=LL(A)
15375 R(P+6)=IA(P,1)+6:P1=P+6:PRINT P"   ";:GOSUB 15700:GOSUB 30000:GOSUB 40000
15380 IA(P,2)=1
15385 VTAB 20:CALL -958:HTAB 13:FLASH:PRINT "LECTURA CAMPOS":NORMAL
15390 PRINT D$;O$(0):PRINT D$;O$(1)
15400 FOR J=1 TO NX
15410 PRINT D$;R$(0);J
15420 INPUT A$:INPUT IX%(J)
15470 PRINT D$;R$(1);IX%(J)
15480 FOR K=1 TO A
15490 INPUT A$
15500 NEXT
15505 IX$(J)=LEFT$(A$,IA(P,1)):NEXT
15507 VTAB 20:CALL -958:HTAB 5:FLASH:PRINT "ATENCION! ORDENACION EN CURSO":NORMA
L
15510 GOSUB 500
15515 VTAB 20:CALL -958:HTAB 12:FLASH:PRINT "ESCRITURA INDICE":NORMAL
15518 ONERR GOTO 17000
15520 PRINT D$;O$(P+6)
```



```
15525 PRINT D$;W$(P+6);0:PRINT NX
15530 FOR J=1 TO NX
15540 PRINT D$;W$(P+6);J
15550 PRINT IX$(PI%(6)):PRINT IX%(PI%(J))
15560 NEXT
15570 PRINT D$;C1$
15575 POKE 216,0
15580 GOSUB 42000
15590 RETURN
15700 REM ASIGNACION DE FICHEROS
15710 VTAB 20:INPUT "CORRECTAS RANURA Y UNIDAD? ";A$
15720 FI A$="S" OR A$="" THEN RETURN
15730 VATB 20:CALL -958:PRINT "RANURA "S(P1);
15740 HATB 6:INPUT "";A$:S(P1)=VAL(A$)
15750 VATB 20:HTAB 21:PRINT "UNIDAD";D(P1);
15760 HATB 27:INPUT "";A$:D(P1)=VAL(A$):PRINT
15780 HATB 23:INPUT "MODIFICACIONES? ";A$
15790 IF A$="S" THEN 15730
15800 RETURN
16000 REM CLAVE MULTIPLICACION
16010 K=1
16020 VTAB 14+K:INPUT "Y EL CAMPO? ";A$:B=VAL(A$):CALL -958:IF A$="" THEN 16060
16030 IF B<1 OR B>NC THEN 16020
16040 VTAB 14+K:HTAB 18:PRINT LB$;:HTAB 29:INPUT "LO CONFIRMA? ";A$:IF LEFT$(A$,
1)<>"S" THEN 16020
16050 VT(K)=B:K=K+1:IF K<5 THEN 16020
16060 NI=K-1:VTAB 15+K:CALL -958:INPUT "CONFIRMA TODO? ";A$
16070 OK=1:IF LEFT$(A$,1)="N" THEN OK=0
16099 RETURN
17000 IF PEEK(222)<>9 THEN 63900
17010 VATB 22:HTAB 12:FLASH:PRINT "NO HAY ESPACIO":NORMAL
17020 PRINT :PRINT "PULSE RETURN ";:GET A$
17022 VATB 23:PRINT
17025 GOSUB 15250
17040 RUN
20000 REM CREACION NUEVO FICHERO
20010 HOME:VTAB 9:PRINT "RANURA"S(3)" UNIDAD"D(3):P1=3:GOSUB 15700:FOR I=4 TO 6:
S(I)=S(3):D(1)=D(3):NEXT
20015 GOSUB 30000:GOSUB 40000
20020 NC=0:NM=14
20030 FOR I=1 TO NM
20040 LB$(I)="":TP(I)=0:LL(I)=0
20050 NEXT
20060 GOSUB 63000
20070 REM ESCRITURA FICHERO LINK.CTRL(CONTROL CADENA)
20075 VATB 23:CALL -958:FLASH:HTAB 13:PRINT "PREPARACION":NORMAL
20080 PRINT D$;O$(3)
20090 PRINT D$;W$(3)
20100 PRINT NC
```

```
20110 FOR I=1 TO NC
20120 PRINT LB$(I)","TP(I)","LL(I)
20130 NEXT
20140 PRINT D$;C$(3)
20150 REM INICIALZACION FICHERO ENTRY
20160 PRINT D$;O$(4)
20170 FOR I=0 TO EM
20180 PRINT D$;W$(4);I
20190 PRINT 0
20200 NEXT
20210 PRINT D$;C$(4)
20220 PRINT D$;O$(5)
20230 PRINT D$;W$(5);0
20240 PRINT 1
20250 PRINT D$;C$(5);0
20260 FL=1:DF=0
20270 GOSUB 42000
20280 GOSUB 40000
20285 R(6)=LR:GOSUB 30000
20290 RETURN
25000 REM PARAMETROS IMPRESION
25010 HOME:HATB 12:PRINT "IMPRESORA TIPO:"
25020 VATB 5:HATB 5:IF NOT PT THEN PRINT "EN PARALELO";:GOTO 25030
25025 PRINT "EN SERIE";
25030 HTAB 20:PRINT "DE "CN " COLUMNAS"
25040 VTAB(15): HTAB(20): INPUT "MODIFICACIONES? ": A$: IF LEFT$(A$,1) <> "S"
THEN 25100
25050 VTAB 18:INPUT "IMPRESORA (S/P)? ":PT=0
25060 IF A$="S" THEN PT=1
25070 INPUT "MUMERO COLUMNAS? ";A$:CN=INT (VAL(A$))
25080 GOTO 25000
25100 GOSUB 42000
25110 HOME
25120 PRINT "COLUMNAS DE IMPRESION? ",CN
25130 PRINT
25140 PRINT "LINEAS POR PAGINA? ",E(0)
25150 PRINT:PRINT
25160 PRINT "AT.     CAMPO N. ",E(1)
25170 PRINT "FIRMA   CAMPO N. ",E(2)
25180 PRINT "DIRECCION CAMPO N. ",E(3)
25190 PRINT "CODIGO POSTAL CAMPO N. ",E(4)
25200 PRINT "CIUDAD CAMPO N. ",E(5)
25210 PRINT
25220 PRINT "ALTURA ETIQUETA",E(6)
25230 PRINT "ANCHURA ETIQUETA",E(7)
25240 PRINT "MARGEN SUPERIOR",E(8)
25250 PRINT "MARGEN IZQUIERDO",E(9)
25260 PRINT:PRINT :PRINT
25270 HATB 20:INPUT  "MODIFICACIONES? ";A$
```



```
25280 A$=LEFT$(A$,1)
25290 IF A$="N" OR A$="" THEN 26000
25300 FOR I=0 TO 9
25310 VTAB 3+I+2*(I>0)+(I>5):HTAB 33
25320 INPUT "";A$:IF A$<>"" THEN E(I)=INT (VAL(A$))
25325 VTAB 3+I+2*(I>0)+(I>5):HTAB 33:PRINT E(I)"   "
25330 NEXT
25340 IF E(6)<E(8)+4 THEN 25300
25350 IF E(7)<E(9)+20 THEN 25300
25390 GOTO 25110
26000 GOSUB 30000
26010 RETURN
30000 REM ESCRITURA DEL SIST.CTRL (CONTROL DEL SISTEMA).
30010 PRINT D$;O1$"SIST.CTRL,S6,D1"
30020 PRINT D$;W1$"SIST.CTRL"
30030 PRINT NC","EM
30035 FOR I=0 TO 9:PRINT E(I):NEXT
30040 FOR I=1 TO NC:PRINT LB$(I)","TP(I)","LL(I):NEXT
30050 RESTORE
30060 FOR I=0 TO 10:READ A$:PRINT A$","R(I)","S(I)","D(I):NEXT
30070 PRINT D$;C1$"SIST.CTRL"
30080 RETURN
35000 REM LECTURA FICHERO BD.CTRL
35010 PRINT D$;O$(2)
35020 PRINT D$;R$(2)
35030 INPUT IA(I,0),IA(I,1)IA(I,2)
35040 FOR I=1 TO 4
35060 NEXT
35070 PRINT D$;C$(2)
35080 RETURN
40000 REM LECTURA DEL FICHERO SITS.CTRL
40010 B$(0)="":B$(1)=",R"
40020 PRINT D$;1$"SIST.CTRL,S6,D1"
40030 PRINT D$;R1$"SIST.CTRL"
40040 INPUT NC,EM
40045 FOR I=0 TO 9:INPUT E(I):NEXT
40050 FOR I=1 TO NC:INPUT LB$(I),TP(I),LL(I):NEXT
40060 FOR I=0 TO 10
40070 L$="":INPUT A$,R(I),S(I),D(I):IF R(I) THEN L$=",L"+STR$(R(I))
40080 D$(I)=O1$+A$+L$+",S"+STR$(S(I))+",D"+STR$(D(I))
40090 R$(I)=R1$+A$+B$(R(I)>0)
40100 W$(I)=W1$+A$+B$(R(I)>0)
40110 C$(I)=C1$+A$
40120 NEXT
40130 PRINT D$;C1$"SIST.CTRL"
40140 RETURN
42000 REM ESCRITURA DEL FICHERO BD.CTRL (CONTROL BASE DE DATOS).
42010 PRINT D$;D$(2)
42020 PRINT D$;W$(2)
```

```
42030 PRINT NX","EA","FL","PT","CN","DF
42040 FOR I=1 TO 4
42050 PRINT IA(I,0)","IA(I,1)",IA(I,2)
42060 NEXT
42070 PRINT D$;C$(2)
42080 RETURN
60000 MENU PRINCIPAL PROGRAMA CONFIGURACION DEL SISTEMA
60010 GOSUB 62000
60020 ONERR GOTO 60050
60030 GOSUB 40000:GOSUB 35000
60035 POKE 216,0
60040 GOTO 60500
60050 REM INICIALIZACION
60060 IF PEEK(222)<>5 THEN 63900
60070 POKE 216,0
60075 NEXT
60090 GOSUB 62500
60095 R(0)=LL(1)+6:R(1)=LR
60100 GOSUB 30000
60110 GOSUB 40000
60120 GOSUB 42000
60150 REM ESCRITURA FICHEROS MAESTROS
60155 VTAB 22:CALL -958:HTAB 4:INPUT "CUANTOS REGISTROS? ";A$:A=VAL(A$)
60158 HTAB 8:FLASH:PRINT "COMPRUEBO SI ESTAN":NORMAL
60160 ONERR GOTO 60360
60190 PRINT D$;O$(0)
60200 PRINT D$;O$(1)
60205 PRINT D$W$(0)0:PRINT "":PRINT 0:PRINT D$W$(1)0:PRINT ""
60210 FOR I=1 TO A
60220 PRINT D$;W$(0);I
60230 PRINT FR$;PRINT I
60240 PRINT D$;W$(1);I
60250 PRINT ST$
60260 NEXT
60270 I=I+1
60280 PRINT D$;C1$:EM=1
60290 VTAB 23:CALL -958:HTAB 1:PRINT "NUMERO MAXIMO DE REGISTROS: ";EM;
60300 HTAB 26:INPUT "MODIFICACIONES? ";A$
60310 IF LEFT$(A$,1)="N" THEN GOSUB 30000:GOSUB 42000:GOTO 60500
60320 IF LEFT$(A$,1)<>"S" THEN 60290
60330 PRINT D$"ELIMINACION FICHERO MST.INX"
60340 PRINT D$"ELIMINACION FICHERO MST.FL"
60350 GOTO 60090
60360 IF PEEK(222)<>9 THEN 63900
60370 ONERR GOTO 60460
60380 PRINT D$;O$(0)
60390 PRINT D$;O$(1)
60400 I=I+1
60410 PRINT D$;R$(0);I
```



```
60420 INPUT A$
60430 PRINT D$;R$(1);I
60440 INPUT A$
60450 POKE 216,0:GOTO 60280
60460 IF PEEK(222)<>5 THEN 63900
60470 PRINT D$;C1$
60480 GOTO 60380
60500 REM MENU PRINCIPAL PROGRAMA CONFIGURACION SISTEMA
60505 DIM IX$(EM),I%(EM),PI%(EM)
60510 HOME:HTAB 10:PRINT "RECONFIGURACION SISTEMAS"
60520 VTAB 5
60530 PRINT " 1- ACTUALIZACION INDICES ADICIONALES"
60540 PRINT :PRINT " 2- CREACION ARCHIVOS CONCATENADOS"
60550 PRINT :PRINT " 3- CONFIGURACION IMPRESORA"
60560 PRINT :PRINT " 4- RETORNO AL MENU PRINCIPAL"
60570 VTAB 22:HTAB 20:PRINT "B.B.I."
60600 VTAB 15:HTAB 10:INPUT "CUAL? ";A$
60610 I=VAL(A$)
60620 IF I<1 OR I>4 THEN 60510
60630 ON GOSUB 15000,20000,25000,61000
60640 GOTO 60510
61000 REM RETORNO A LA BASE DE DATOS
61010 PRINT D$,C1$
61020 PRINT D$"RUNLOGO,S6,D1"
62000 REM INICIALIZACION VARIABLES
62010 D$=CHR$(4)
62020 T$= "              "
62030 NM=14
62040 DIM TP(NM),LL(NM),LB$(NM),LD(1),ID(4,2),V1$(NM),V2$(NM)
62050 FR$=CHR$(95)
62100 D1$="OPEN"
62110 R1$="READ"
62120 W1$="WRITE"
62130 C1$="CLOSE"
62160 RETURN
62500 REM PRIMERA VEZ
62510 HOME:CN=80:RESTORE
62520 FOR I=0 TO 10:READ A$:NEXT
62530 FOR I=0 TO 10:READ R(I):NEXT
62540 FOR I=0 TO 10:READ S(I):NEXT
62550 FOR I=0 TO 10:READ D(I):NEXT
62555 FOR I=0 TO 9:READ E(I):NEXT
62560 VATB 9:HTAB 9:INPUT "BASE DE DATOS EN UNIDAD:";A$:A=VAL(A$)
62570 IF A<>1 AND A<>2 THEN 62560
62580 PRINT :HATB 22:INPUT "SEGURO? ";A$
62590 IF LEFT$(A$,1)<>"S" THEN 62560
62600 FOR I=0 TO2:D(I)=A:NEXT
63000 REM CONFIGURACION DEL SISTEMA
63020 HOME:HTAB 18:PRINT "CONFIGURACION DEL SISTEMA"
```

```
63030 HTAB 13:PRINT "CUANTOS CAMPOS?: ";:IF NC THEN PRINT NC;
63040 IF NC>NM OR NC<1 THEN 63020
63050 VATB 2:HTAB 27:CALL -868:PRINT NC
63060 VATB 4:HTAB 11:PRINT "NOMBRE";:HTAB 21:PRINT "TIPO (A/N)";:HTAB 25:PRINT "
      LONGITUD"
63070 FOR I=1 TO NC
63080 VATB I+5:CALL -868:IF I>10 THEN HTAB 2
63090 PRINT I;"CAMPO";
63100 HTAB 10:IF LB$(I)<>"" THEN PRINT LB$(I);
63110 HTAB 10:INPUT "";A$
63113 IF A$="" AND LB$(I)="" THEN 63080
63116 IF A$<>"" THEN LB$(I)=A$
63120 VTAB I+5:HTAB 10:PRINT LB$(I)
63140 IF TP(I)=1 THEN INPUT "ALFANUMERICO ";A$
63145 IF TP(I)=2 THEN INPUT "  NUMERICO ";A$
63150 IF TP(I)=0 THEN INPUT " ";A$
63160 VTAB I+5:HTAB 20
63170 IF A$="A" THEN TP(I)=1:CALL -868:PRINT "ALFANUMERICO":GOTO 63200
63180 IF A$="N" THEN TP(I)=2:CALL -868:PRINT "   NUMERICO ":GOTO 63200
63190 IF A$<>"" OR TP(I)=0 THEN 63130
63200 VTAB I+5:HTAB 35:CALL -868
63210 IF LL(I)<>0 THEN PRINT LEFT$(T$,4-LEN(STR$(LL(I))));LL(I);:HTAB 35
63220 INPUT "";A$
63230 IF A$="" AND NOT LL(I) THEN 63200
63240 IF A$<>"" THEN LL(I)=INT (VAL(LEFT$(A$,3)))
63250 IF LL(I)<1 OR LL(I)>29 THEN 63200
63260 VTAB I+5:HTAB 35:CALL -868:PRINT LEFT$(T$,4-LEN(STR$(LL(I))));LL(I)
63270 NEXT
63280 LR=NC:FOR I=1 TO NC:LR=LR+LL(I):NEXT
63290 VTAB 21:PRINT "LONGITUD REGISTRO "LR" BYTES"
63300 VTAB 23:HTAB 20:INPUT "MODIFICAR? ";A$
63310 IF LEFT$(A$,1)<>"N" THEN 63000
63320 ST$=""
63330 J=LR-1:IFJ>255 THEN J=255
63340 FOR I=1 TO J:ST$=ST$+FR$:NEXT
63350 RETURN
63800 DATA "MST.INX","MST.FL","DB.CTRL","LNK.CTRL","LNK.ENTRY","LNK.PTR","LNK.FR
","INDEX.1","INDEX.2","INDEX.3","INDEX.4"
63810 DATA 1,1,0,0,5,5,1,1,1,1,1
63820 DATA 6,6,6,6,6,6,6,6,6,6,6
63830 DATA 1,1,1,2,2,2,2,2,2,2,2
63840 DATA 66,0,0,0,0,0,6,39,1,4
63900 HOME:FLASH:VTAB 10:HTAB 12:PRINT "CODIGO ERROR = "PEEK(222):NORMAL:END
```


Programa de chequeo

```
10 GOTO 60000
100 REM ****************************
101 REM **    A P P L E   I I     **
102 REM **                        **
103 REM ** BASE DE DATOS PERSONAL **
104 REM ****************************
200 GR:ER=0
210 FOR I=1 TO EM
220 PRINT R$(0);I:INPUT A$:INPUT A
225 IF(1<=EA) AND (A$<>FR$) THEN PRINT R$(1)A:INPUT AA$:IF A$<>AA$ THEN ER=1
230 I%(A)=I%(A)+1
240 COLOR =I%(A):IF I%(A)>15 THEN COLOR =15
250 G=A:IF EM>1599 THEN G=INT(1599*A/EM)
260 Y=INT (G/40):X=G-Y*40:PLOT X,Y
270 NEXT :PRINT D$
300 IF I%(0)>0 THEN ER=1
310 FOR I=1 TO EM :IF I%(I)<>1 THEN ER=1
320 NEXT
330 RETURN
1000 REM LECTURA DEL FICHERO SIST.CTRL
1010 B$(0)="":B$(1)=" ,R"
1020 PRINT O1$"SIST.CTRL,S6,D1"
1030 PRINT R1$"SIST.CTRL":INPUT NC(0),EM
1040 FOR I=0 TO 9:INPUT E(1):NEXT :LP=E(0)
1050 FOR I=0 TO NC(0):INPUT LB$(I,0),TP(I,0),LL(I,0):NEXT
1060 FOR I=0 TO 10
1070 L$="":INPUT A$,R(I),S(I),D(I):IF R(I) THEN L$=" ,L"+STR$(R(I))
1080 D$(I)= O1$+A$+L$+" ,S"+STR$(S(I))+" ,D"+STR$(D(I))
1090 R$(I)= R1$+A$+B$(R(I)>0)
1100 W$(I)= W1$+A$+B$(R(I)>0)
1110 C$(I)= C1$+A$
1120 NEXT
1130 PRINT C1$"SIST.CTRL"
1140 RETURN
1500 REM LECTURA DEL FICHERO BD.CTRL
1510 PRINT O$(2)
1520 PRINT R$(2):INPUT NX,EA,PL,PT,CN,DF
1540 FOR I=1 TO 4:INPUT IA(I,0),IA(I,1),IA(I,2):NEXT
1570 PRINT C$(2)
1580 RETURN
2000 REM HEAPSORT
2005 FOR L=0 TO EA:I%(L)=L:NEXT
2010 IF EA=1 THEN RETURN
2015 L=INT(EA/2)+1:R=EA
2020 IF L>1 THEN L=L-1:A=I%(L):GOTO 2040
2025 A=I%(R):I%(R)=I%(1):R=R-1
```

```
2030 IF R=1 THEN I%(1)=A:RETURN
2040 J=L
2045 I=J:J=2*J
2050 IF J>R THEN I%(I)=A:GOTO 2020
2055 IF J<R THEN PRINT R$(1);I%(J):INPUT A$:PRINT R$(1),I%(J+1):INPUT B$:IF A$<B
     $ THEN J=J+1
2060 PRINT R$(1);A:INPUT A$:PRINT R$(1),I%(J):INPUT B$:IF A$<B$ THEN I%(I)=I
     %(J):GOTO 2045
2065 I%(I)=A:GOTO 2020
20000 REM COMPROBACION FICHERO MAESTRO
20010 GOSUB 62000:GOSUB 1000:GOSUB 1500
20020 DIM I%(EM)
20030 PRINT D$(0):PRINT D$(1)
20100 GOSUB 200
20250 IF ER THEN 21000
20290 HOME:PRINT C1$
20300 VTAB 21:HTAB 12:PRINT "TODO CORRECTO !"
20310 PRINT :HTAB 12:INPUT "PULSE RETURN";A$
20320 RETURN
21000 REM MODIF. 1'NIV
21005 GOSUB 50000:IF NOT C THEN PRINT C1$:RETURN
21007 HOME :VTAB 22:HTAB 10:PRINT "INTENTO DE CORRECION"
21010 PRINT W$(0),0:PRINT "":PRINT 0
21020 IF EA<EM THEN FOR I=EA+1 TO EM:PRINT W$(0);I:PRINT FR$:PRINT I:NEXT
21090 FOR I=0 TO EM:I%(I)=0:NEXT
21100 GOSUB 200:IF NOT ER THEN 20290
21200 GOSUB 200
21300 FOR I=1 TO EA
21310 PRINT R$(1)I%(I):INPUT A%
21320 PRINT W$(0)I:PRINT I%(I)
21330 NEXT :PRINT D$
21400 FOR I=1 TO EM:I%(I)=0:NEXT
21410 GOSUB 200:IF NOT ER THEN 20290
21500 HOME :VTAB 21:HTAB 10:PRINT "IMPOSIBLE SUBSANAR"
21510 PRINT :HTAB 13:INPUT  "(PULSE RETURN)";A$
21520 PRINT C1$
21530 RETURN
40000 REM COMPROBACION CADENA
40010 GOSUB 62000:GOSUB 1000:GOSUB 1500
40020 IF NOT PL THEN RETURN
40030 PRINT D$(4):PRINT D$(5)
40040 PRINT R$(5):INPUT DL
40050 DL=DL-1
40060 DIM V%(DL):GR
40100 FOR I=0 TO EM
40110 PRINT R$(4)I:INPUT P
40120 IF P THEN GOSUB 40300
40130 NEXT
40140 ER=0
```



```
40150 FOR I=0 TO DL
40160 IF V%(I)<>1 THEN IF I THEN ER=1
40170 NEXT :PRINT D$
40180 IF NOT ER THEN 20290
40200 GOSUB 50000:IF NOT C THEN PRINT C1$:RETURN
40210 HOME:VTAB 22:HTAB 10:PRINT "INTENTO DE CORRECCION"
40220 GOTO 40500
40300 V%(P)=V%(P)+1
40302 COLOR =V%(P):IF V%(P)>15 THEN COLOR =15
40304 G=P:IF DL>1599 THEN G=INT(1599*P/DL)
40306 Y=INT(G/40):X=G-Y*40:PLOT X,Y
40310 PRINT R$(5)P:INPUT P
40320 IF P THEN 40300
40340 RETURN
40500 PRINT R$(4)0:INPUT IL
40505 L=IL
40510 FOR I=1 TO DL
40520 IF V%(I)=0 THEN PRINT W$(5)I:PRINT IL:L=I:V%(I)=1
40530 NEXT
40550 FOR I=L TO DL
40560 IF V%(I)=0 THEN PRINT W$(5)I:PRINT L:L=1
40570 NEXT
40600 PRINT W$(4)0:PRINT L
40610 PRINT D$
40700 HOME:PRINT C1$
40720 VTAB 21:HTAB 10:PRINT "VOLVER A INTENTAR VERIFICACION"
40730 HTAB 10:PRINT "NO PUEDO HACER OTRO"
40740 HTAB 10:INPUT "PULSE RETURN";A$
40750 RETURN
50000 HOME:VTAB 21:HTAB 15:FLASH:PRINT "ERROR":NORMAL
50010 HTAB 1:PRINT "SE RECOMIENDA UTILIZAR DISCOS COPIA"
50020 HTAB 2:PRINT "PULSE (F) PARA TERMINAR"
50030 HTAB 2:PRINT "PULSE (C) PARA INTENTAR LA CORRECCION"
50040 PRINT :C=(A$="C")
50050 IF A$<>"F" AND NOT C THEN 50000
50060 HOME
50199 RETURN
60000 REM MENU PRINCIPAL
60100 TEXT:HOME:CLEAR
60110 VTAB 6:HTAB 4:PRINT " 1-_VERIFICAR ARCHIVO 1"
60120 VTAB 8:HTAB 4:PRINT " 2- VERIFICAR ARCHIVO 2"
60130 VTAB 10:HTAB 4:PRINT " 3- MENU PRINCIPAL"
60140 VTAB 12:HTAB 4:PRINT " 4- FIN"
60200 VTAB 23:HTAB 20:PRINT " B.B.I."
60210 VTAB 16:HTAB 10:INPUT "CUAL? ";A$
60220 ON VAL(A$) GOSUB 20000,40000,60400,60300
60230 GOTO 60000
60300 REM FIN
60310 PRINT C1$
```

```
60320 PRINT "LOCKLOGO,D1"
60330 PRINT "MAXFILES 3"
60340 HOME:PRINT "ADIOS !"
60350 END
60400 REM VOLVER AL MENÚ PRINCIPAL
60410 PRINT "RUNLOGO,D1"
62000 REM INICIALIZACION VARIABLES
62010 I=0:T1$="-":FOR J=1 TO 4:T1$=T1$+T1$:NEXT
62020 D$=CHR$(4):FR$=CHR$(95)
62030 NM=15
62040 DIM LB$(NM,1),TP(NM,1),LL(NM,1),RB$(NM,1)RV$(NM,1),EY$(25)
62050 O1$=D$+"OPEN":R1$=D$+"READ":V1$=D$+"WRITE":C1$=D$+"CLOSE"
62060 DIM MI$(NM,1),MA$(NM,1)
62070 FOR I=1 TO NM:FOR J=0 TO 1:MA$(I,J)=FR$:LB$(NM,J)="TOTAL":NEXT J,I
62200 RETURN
```

En anteriores volúmenes de la Biblioteca Básica Informática nos hemos acercado ya al manejo y almacenamiento de la información de una forma informatizada. Hemos visto también cómo se lograba esto mediante un programa comercial (el dBASE).

Cuando intentamos aplicar estos conocimientos a un ordenador personal nos encontramos, sin embargo, con una serie de limitaciones (como la memoria) que impiden un aprovechamiento óptimo de nuestro banco de datos. En este volumen veremos cómo diseñar nuestra propia base de datos para adaptarla expresamente a nuestras necesidades.

395 pts.
(incluido IVA)